Destroy this report when no longer needed. Do not return it to the originator.

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

The contents of this report are not to be used for advertising, publication, or promotional purposes. Citation of trade names does not constitute an official endorsement or approval of the use of such commercial products

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1991	Final report

4. TITLE AND SUBTITLE
Application of Finite Element, Grid Generation, and Scientific Visualization Techniques to 2-D and 3-D Seepage and Groundwater Modeling

5. FUNDING NUMBERS

6. AUTHOR(S)
Fred T. Tracy

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
USAE Waterways Experiment Station, Information Technology Laboratory, 3909 Halls Ferry Road, Vicksburg, MS 39180-6199

8. PERFORMING ORGANIZATION REPORT NUMBER
Technical Report ITL-91-3

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Assistant Secretary of the Army (R&D)
Washington, DC 20315

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Available from National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This technical report describes new advances in the computational modeling of groundwater and seepage using the finite element method (FEM) in conjunction with tools and techniques typically used by the aerospace engineers. The unsolved environmental issues regarding our hazardous and toxic waste problems must be resolved, and significant resources must be placed on this effort. Some military bases are contaminated with hazardous waste that has entered the groundwater domain. A groundwater model that takes into account contaminant flow is therefore critical. First, an extension of the technique of generating an orthogonal structured grid (using the Cauchy-Riemann equations) to automatically generate a flow net for two-dimensional (2-D) steady-state seepage problems is presented for various boundary conditions. Second, a complete implementation of a three-dimensional (3-D) seepage package is described where (1) grid generation is accomplished using the EAGLE program, (2) the seepage and groundwater analysis for either confined or unconfined steady-state flow, homogeneous or inhomogeneous

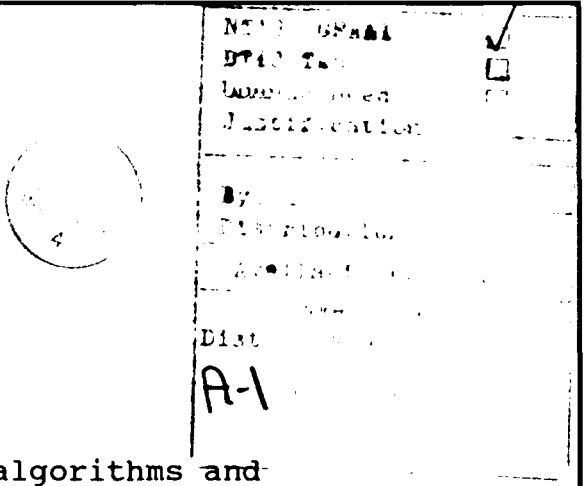
(Continued)

14. SUBJECT TERMS		15. NUMBER OF PAGES
Finite element method	Scientific visualization	228
Flow net	Seepage	16. PRICE CODE
Groundwater		

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED		

13. ABSTRACT (Concluded).

media, and isotropic or anisotropic soil is accomplished with no restriction on the FE grid or requirement of an initial guess of the free surface for unconfined flow problems, and (3) scientific visualization is accomplished using the program FAST developed by NASA. A primary aspect of this report is a description of the developed complex computational techniques required to achieve the 3-D model. Finally, examples showing both generated flow nets for 2-D problems and results for both theoretical and practical 3-D problems are presented.



PREFACE

This report documents computational algorithms and developed computer programs for modeling two-dimensional (2-D) and three-dimensional (3-D) seepage under dams and groundwater flow in aquifers. A method for automatically generating flow nets for 2-D applications and Cray YMP, scientific visualization, and numerics for 3-D problems are emphasized. Specifically, techniques typically used by aerospace engineers are applied to flow through porous media. This research and development was done and this report was written at the US Army Engineer Waterways Experiment Station (WES), Information Technology Laboratory (ITL), Interdisciplinary Research Group, Computer-Aided Engineering Division (CAED), by Dr. Fred T. Tracy in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Mississippi State University. The work was sponsored by funds provided by the In-House Laboratory Independent Research Program (ILIR) and the Engineering Division, Civil Works Directorate, Headquarters, US Army Corps of Engineers (HQUSACE), under the Computer-Aided Structural Engineering (CASE) Project.

Mr. Leonard I. Huskey is manager of the ILIR Program, and overall coordination is provided by Ms. Mary K. Vincent,

Chief, Office of Technical Programs and Plans, WES. Mr. Donald R. Dressler, Chief, Structures Branch, Engineering Division, is the HQUSACE point of contact for CASE. Mr. H. Wayne Jones, Chief, Engineering and Scientific Applications Center, CAED, is the CASE program manager. The Scientific Visualization Center (SVC), ITL, whose project manager is Mr. Bradley M. Comes, contributed greatly to the success of this undertaking. The work was accomplished at WES under the supervision of Dr. N. Radhakrishnan, Chief, ITL, and Mr. Paul K. Senter, Chief, CAED.

Commander and Director at WES is COL Larry B. Fulton, EN. Technical Director is Dr. Robert W. Whalin.

TABLE OF CONTENTS

		Page
PREFACE		i
LIST OF FIGURES		vii
LIST OF SYMBOLS		xi
CHAPTER		
I.	INTRODUCTION	1
	Introduction	1
	Scope of Dissertation	1
	Two-Dimensional (2-D) Improvements	2
	Three-Dimensional Model	2
	Grid Generation	3
	Conversion and Data Completion	3
	FEM Seepage Analysis	3
	Scientific Visualization	4
	Test Problems	4
	Previous Work	4
II.	GOVERNING EQUATIONS	10
	Flow Equations	10
	Steady-State Solution	12
III.	COMPUTER GENERATED FLOW NETS	14
	Introduction	14
	Governing Equations and Basic Approach	15
	Modified Cauchy-Riemann Conditions	17
	Examples	19
	Four-Sided Regions	19
	Confined Flow under a Weir	19
	Partially Penetrating Slot	22
	Head specified at the slot	23
	Constant discharge velocity specified at the slot	23

Dupuit's Problem 26
Anisotropic Soil 30
Multilayered Problems 32
Axisymmetric Case 35
Comparison with Other Work 37

IV. THREE-DIMENSIONAL NUMERICS 40

Introduction 40
Finite Volume versus Finite Element Comparison 40
General Finite Element Method Formulation . . . 47
Element Formulation 47
Element Shape 48
Isoparametric Element 49
Stiffness Matrix Formulation 51
Numerical Integration 53
Decomposition 54
Discharge Velocity 55
Assemblage with Boundary Conditions . . . 55
Unconfined Flow Problem 58
Description of the Problem 58
Geometric Considerations 59
Reshaped Elements 59
Exit Point 60
Computational Procedure 61
Compute and Store $[K]_{a0}$ for Each Element 62
Assemble $[K]_u$ and $[K]_{G0}$ 62
Solve $[K]_{G0}\{h\}_{G0} = \{Q\}_{G0}$ for $\{h\}_{G0}$. 62
Compute $\{Q\}_u = [K]_u\{h\}_{G0}$ 63
Check for Convergence 63
Compute $[K]_{aj}$ for Each Element . . . 64
Compute $\{\Delta Q\}_{ej}$ and $\{\Delta Q\}_{Gj}$ 65
Compute Different Global Stiffness Matrix $[K]_{G'j}$ 65
Compute $\{\Delta h\}_{Gj}$ 68
Update $\{h\}_{Gj}$ 68
Determine Final Position of Free Surface 69
Compute Final Element Discharge Velocities 77

V. THREE-DIMENSIONAL SEEPAGE PACKAGE AND GROUNDWATER MODEL 83

Introduction 83
Grid Generation 83

Conversion to FEM Format 84
Getting Data 84
Applying Boundary Conditions 84
Combining Different Blocks 87
Applying Bandwidth Minimization. 87
Writing an Output File 88
Cray Version of FEM Program 88
Scientific Visualization 89
Conversion from Element Data to Node Data 89
Conversion from Finite Element to Finite Volume Format 91
Visualization Techniques 93
Initial and Final Grid 94
Isolevels 94
Color Contours on a Surface 95
Particle Traces 95

VI. 3-D GROUNDWATER FLOW IN AN AQUIFER 96

Introduction 96
Groundwater Flow in an Aquifer 96
General Description of Problem 97
Simplified Problem 97
Description of Simplified Problem 97
Analytic Solution for a Partially Penetrating Well 99
Well in an Aquifer 102
Multiple Wells in an Aquifer 105
Quality of Grids Used 105
Solution for a Partially Penetrating Well 106
Grid technique 106
O grid with plug 106
Spacing 107
Analysis of results 108
Visualization 108
Error analysis 117
Solution of Simplified Problem . . . 123
Analysis of results 127
Visualization 127
Error analysis 135
Real-World Example 135
Description of Problem 135
FEM Grid 144
Presentation of Results 147

VII. SUMMARY AND CONCLUSION 160

2-D Flow Nets 160
3-D Modeling 160

REFERENCES 162

APPENDIX

A. FREE SURFACE SUBROUTINES 167

Subroutine FREES 169
Subroutine FSPT 172
Subroutine FIXEL 174

B. CONVERSION TO FEM FORMAT 179

MAIN Program 181
Subroutine ADJAC 183
Subroutine BANMIN 185
Subroutine BC 189
Subroutine COMBIN 192
Subroutine DONE 197
Subroutine OUTFEM 197
Function NONBLK 199

C. CONVERSION TO FAST FORMAT 201

D. GRID GENERATION DATA 209

Surface Generation Data 211
Grid Generation Data 220
FEM Grid Preparation Data 221
FEM Grid 224

LIST OF FIGURES

Figure		Page
1.	Earth Dam with Different Soil Properties	6
2.	FE Grid with Free Surface	8
3.	Earth Dam	10
4.	Tangent-Normal System	17
5.	Weir Problem	20
6.	Weir Problem with Sheet Piles	22
7.	Flow Net for Weir Problem	23
8.	Partially Penetrating Slot	23
9.	Specified Discharge Velocity	24
10.	Dupuit's Problem	26
11.	Results for Dupuit's Problem	29
12.	Earth Dam Problem	30
13.	Weir on Anisotropic Soil	31
14.	Isotropic Results	32
15.	Anisotropic Results	32
16.	Flow across Boundary	33
17.	Three-Layered Problem	35
18.	Results for Three-Layered Problem	36
19.	Fully Penetrating Well "Flow Net"	36
20.	Confined Flow in an Anisotropic Medium	38

Figure Page

21. Author's Results 38

22. Unconfined Flow through Two-Zoned Dam 39

23. Author's Results 39

24. One-Dimensional Seepage Flow 41

25. FEM Version 43

26. Finite Element Type 48

27. Other Shapes 49

28. Unconfined Flow Problem 58

29. FEM Grid with Free Surface 60

30. Exit Point E 61

31. Free Surface through 2-D Element 64

32. Surface of Seepage 66

33. 2-D Exit Point Analysis 67

34. Free Surface across Grid 70

35. Seven-Sided Piece 71

36. Line Segment AB 72

37. Exit Point Computation 73

38. 3-D Free Surface 75

39. Exit Face 76

40. New Element Shapes 78

41. Three-Step Process 78

42. 3-D Version 79

43. Isolated Piece 81

44. Three Steps for Isolated Piece 81

Figure Page

45. Quadrilateral Earth Dam 86

46. Element to Node Conversion 90

47. Block 91

48. Aquifer with Two Wells 98

49. Partially Penetrating Well 100

50. Method of Images 102

51. O Type Grid 107

52. Plug 108

53. Radial Spacing 109

54. Depth Spacing 110

55. Constant I Color Contour Plot 111

56. Constant J Color Contour Plot 113

57. Constant K Color Contour Plot 115

58. Isolevel Surfaces Plot 119

59. Particle Trace Plot 121

60. Percent Error Plot 125

61. Two-Well Problem 127

62. Large Algebraic Grid at Well 128

63. J Surface Color Contour Plot 129

64. I Surface Color Contour Plot 131

65. K Surface Color Contour Plot 133

66. Isolevel Plot 137

67. Flow Lines Plot for One Well 139

68. Elliptic Grid 141

Figure Page

69. Plan View of Aquifer 142

70. Permeability Orientation 143

71. Subregions 145

72. Plan View 148

73. Perspective View 149

74. Grid at Well 151

75. Grid at Impervious Wall 152

76. Homogeneous, Isotropic Medium 153

77. Actual Medium 157

78. Free Surface at Well 159

79. Surface Generation Line Numbers 214

80. Topology for Grid Program 221

LIST OF SYMBOLS

Symbol	Description
A	Angle of rotation.
α	Normalized r for the solution to a partially penetrating well.
α_w	Normalized r_w for the solution to a partially penetrating well.
α_1	Angle between a flow line and the normal to the boundary for material type No. 1.
α_2	Angle between a flow line and the normal to the boundary for material type No. 2.
b	Penetration of a partially penetrating well.
[B]	Matrix relating the gradient to the nodal heads for an element.
$[B]_0$	[B] matrix at the average of the respective (x, y, z) coordinates of the corner nodes of an element.
β	Normalized penetration for the solution to a partially penetrating well.
c	Constant to be evaluated.
c_m	Change in slope.
D	Weighing factor for "one over distance squared" formulation.
d_{An}	Distance between point A and node n.
d_{Bn}	Distance between point B and node n.
d_{Cn}	Distance between point C and node n.

Symbol	Description
$\{\Delta h\}_{Gj}$	Change in global head vector at the jth iteration for unconfined flow problems.
$\{\Delta h\}_{max}$	The maximum change in head for any node from the results of the previous iteration for unconfined flow problems.
ΔQ	Change in flow.
$\Delta Q'$	Modified change in flow.
$\{\Delta Q\}_{ej}$	Change in nodal flows for a given element at the jth iteration for unconfined flow problems.
$\{\Delta Q\}_{Gj}$	Change in global flow vector at the jth iteration for unconfined flow problems.
Δx	Small increment of x.
Δy	Small increment of y.
E	An expression of partial derivatives of shape functions used to compute flows from a specified normal component of discharge velocity.
ϵ	Small number representing the convergence criteria for unconfined flow problems.
f	Shape factor.
F	An expression of partial derivatives of shape functions used to compute flows from a specified normal component of discharge velocity.
$\{F\}$	Nodal forces vector.
G	An expression of partial derivatives of shape functions used to compute flows from a specified normal component of discharge velocity.
γ	Parameter between one-half and one used in the exit line computation.

Symbol	Description
$\Gamma(x)$	Gamma Function.
h	Total head or potential.
$\{h\}$	Nodal head vector.
h_d	Downstream head.
h_D	Datum.
$\{h\}_e$	Nodal head vector for an element.
$\{h\}_{e,j-1}$	Nodal head vector for a given element at the j - 1th iteration for unconfined flow problems.
$\{h\}_G$	Global head vector.
$\{h\}_{Gj}$	Global head vector at the jth iteration for unconfined flow problems.
$\{h\}_{G,j-1}$	Global head vector at the j - 1th iteration for unconfined flow problems.
$\{h\}_{G0}$	Global head vector computed before the first iteration for unconfined flow problems.
$H_H(t)$	Head water level at time t.
h_i	Head for finite volume cell i.
h_{i-1}	Head for finite volume cell i - 1.
h_{i+1}	Head for finite volume cell i + 1.
h_p	Pressure head.
h_{pA}	Pressure head at point A.
h_{pB}	Pressure head at point B.
$H_T(t)$	Tail water level at time t.
h_u	Upstream head.

Symbol	Description
h_1	Head for the first node of an element.
H_1	Constant value of head.
h_2	Head for the second node of an element.
H_2	Constant value of head.
h_9	Head for the ninth node of an element.
$\{h\}_8$	Element head vector, excluding the internal node.
$\{h\}_9$	Element head vector, including the internal node.
h_9'	Modified head for the ninth node of an element.
i	Gradient.
$\{i\}$	Gradient vector.
J	Jacobian for a one-dimensional element.
[J]	Jacobian matrix for a three-dimensional element.
k	Scalar permeability.
[k]	3 by 3 permeability matrix.
[K]	Stiffness matrix.
$[K]_a$	Actual element stiffness matrix used.
$[K]_{ai}$	Actual stiffness matrix used for the ith element.
$[K]_{aj}$	Actual stiffness matrix for a given element for the jth iteration for unconfined flow problems.
$[K]_{a0}$	Actual stiffness matrix used for an element before the first iteration for unconfined flow problems.

Symbol	Description
$[K]_e$	Element stiffness matrix.
$[K]_G$	Global stiffness matrix.
$[K]_{G0}$	Global stiffness matrix after being modified by boundary conditions before the first iteration for unconfined flow problems.
$[K]_{G'j}$	Alternate global stiffness matrix for the jth iteration for unconfined flow problems.
k_H	Horizontal component of permeability.
k_{p1}	Permeability for material type No. 1.
k_{p2}	Permeability for material type No. 2.
k_{p3}	Permeability for material type No. 3.
$[K]_u$	Global stiffness matrix unmodified by boundary conditions.
k_V	Vertical component of permeability.
$K_0(x)$	Modified Bessel Function of the second kind.
k_1	Component of permeability in the first principal direction.
k_2	Component of permeability in the second principal direction.
k_3	Component of permeability in the third principal direction.
$[K]_8$	8 by 8 element stiffness matrix.
$[K]_9$	9 by 9 element stiffness matrix.
K_{99}	The ninth-row and ninth-column term of the element stiffness matrix.

Symbol	Description
k'	Permeability in the transformed coordinate system.
k_1'	Modified permeability for the stream function computation for material type No. 1.
k_2'	Modified permeability for the stream function computation for material type No. 2.
k_3'	Modified permeability for the stream function computation for material type No. 3.
$\mathbf{K}$	3 by 3 permeability tensor.
l	Length of an aquifer.
m_a	Slope of the first node for the computation of the exit point.
m_b	Slope of the second node for the computation of the exit point.
m_c	Slope of the third node for the computation of the exit point.
n	Porosity or ratio of the volume of voids to the total volume.
N	Normal coordinate of a tangent-normal coordinate system.
$\{N\}$	Shape function vector.
$[N]$	Shape function matrix.
N_e	Number of equipotential drops.
N_E	Number of elements.
N_f	Number of flow paths.
N_i	Shape function for the ith node of an element.

Symbol	Description
N_1	Shape function for the first node of an element.
N_2	Shape function for the second node of an element.
$\{N\}_4$	Shape function vector for the four nodes of a face of a brick element.
$[N]_4$	Shape function matrix for the four nodes of a face of a brick element.
N_9	Shape function for the ninth node of an element.
$[P]$	Matrix of partial derivatives of shape functions.
ϕ	Total head or potential.
ϕ_e	Second Euler angle.
Φ	Complex potential.
ψ	Stream function.
ψ_D	Value of stream function at point D.
ψ_e	Third Euler angle.
ψ_E	Value of stream function at point E.
ψ_I	Value of stream function at point I.
ψ_J	Value of stream function at point J.
ψ_{TI}	Value of partial of stream function with respect to T at point I.
ψ_{TJ}	Value of partial of stream function with respect to T at point J.
ψ_{total}	Total amount of stream function.
ψ_1	Constant value of stream function.

Symbol	Description
ψ_2	Constant value of stream function.
q	Flux density for the partially penetrating well problem.
Q	Quantity of flow.
Q′	Five-component variable for Euler Equations in strong conservative form.
$\{Q\}$	Nodal flow vector.
Q_E	Flow specified at an external node.
$\{Q\}_G$	Global flow vector.
$\{Q\}_{GO}$	Global flow vector computed before the first iteration for unconfined flow problems.
$\{Q\}_u$	Nodal flow vector computed from the unmodified stiffness matrix and the nodal heads.
Q_1	Flow for node 1.
Q_2	Flow for node 2.
$\{Q\}_4$	Element flow vector for the four nodes of a face of a brick element.
$\{Q\}_8$	Element flow vector, excluding the internal node.
$\{Q\}_9$	Element flow vector, including the internal node.
r	Radial distance from the center of the well.
$\{r\}$	Coordinate vector.
$\{r\}_A$	Coordinate vector at point A.
$\{r\}_B$	Coordinate vector at point B.

Symbol	Description
$[r]_e$	Coordinate matrix for an element.
$\{r\}_i$	Coordinate vector at node i.
r_w	Radius of the well.
$\{r\}_0$	Value of coordinate vector where the pressure head is zero.
$[r]_4$	Coordinate matrix for the four nodes of a face of a brick element.
ρ	Density of the soil-water complex.
ϱ	Concentration of water in the soil-water complex.
s	Parameter that varies between zero and one.
S_s	Specific storage.
s_0	Value of s where the pressure head is zero.
s_1	Final value of s in the exit line computation.
t	Time.
T	Tangent coordinate of a tangent-normal coordinate system.
t_w	Thickness of a partially penetrating well.
θ_e	First Euler angle.
θ_1	Angle between an equipotential line and the normal to the boundary for material type No. 1.
θ_2	Angle between an equipotential line and the normal to the boundary for material type No. 2.
u	x component of discharge velocity.
U	Energy.

Symbol	Description
$\{u\}$	Nodal displacements vector.
$u_{i-\frac{1}{2}}$	Discharge velocity at the left flux boundary of a finite volume cell.
$u_{i+\frac{1}{2}}$	Discharge velocity at the right flux boundary of a finite volume cell.
v	y component of discharge velocity.
$\vec{v}$	Velocity vector.
$\{v\}$	Discharge velocity vector.
$\{v\}_A$	Discharge velocity vector at point A.
$\{v\}_B$	Discharge velocity vector at point B.
$\{v\}_C$	Discharge velocity vector at point C.
v_j	jth component of discharge velocity.
$\{v\}_n$	Discharge velocity vector at node n.
v_N	Normal component of discharge velocity.
V_o	Specified discharge velocity.
$\{v\}_0$	Discharge velocity vector at the average of the respective (x, y, z) coordinates of the corner nodes of an element.
v_j	jth component of the actual velocity of the water particles.
w	z component of discharge velocity.
x	x coordinate.
x_a	x coordinate of the first node for the computation of the exit point.
x_b	x coordinate of the second node for the computation of the exit point.

Symbol	Description
x_c	x coordinate of the third node for the computation of the exit point.
$\{x\}_e$	x coordinate vector for an element.
x_i	x coordinate at node i of an element.
x_j	x, y, or z, depending on whether j = 1, 2, or 3.
x_o	x coordinate of a point on a boundary.
(x_0, y_0)	Location of a well in an aquifer.
x_1	x coordinate of the first node of an element.
x_2	x coordinate of the second node of an element.
$\{x\}_4$	x coordinate vector for the four nodes of a face of a brick element.
x_9	x coordinate of the ninth node of an element.
x'	x coordinate in a new coordinate system.
(ξ, η, ζ)	Coordinates for computational type space where finite elements are mapped.
(ξ_i, η_i, ζ_i)	Coordinates for computational type space where finite elements are mapped at node i.
y	y coordinate.
y_a	y coordinate of the first node for the computation of the exit point.
y_b	y coordinate of the second node for the computation of the exit point.
y_c	y coordinate of the third node for the computation of the exit point.

Symbol	Description
y_D	Value of y coordinate at point D.
$\{y\}_e$	y coordinate vector for an element.
y_E	Value of y coordinate at point E.
y_i	y coordinate at node i of an element.
y_o	y coordinate of a point on a boundary.
$\{y\}_4$	y coordinate vector for the four nodes of a face of a brick element.
y_9	y coordinate of the ninth node of an element.
y'	y coordinate in a new coordinate system.
z	z coordinate.
$\{z\}_e$	z coordinate vector for an element.
z_i	z coordinate at node i of an element.
z_{max}	Maximum z value of the nodes of the FE grid.
z_{min}	Minimum z value of the nodes of the FE grid.
z_t	z value at the top of an aquifer.
$\{z\}_4$	z coordinate vector for the four nodes of a face of a brick element.
z_9	z coordinate of the ninth node of an element.
ζ	Normalized z for the solution to a partially penetrating well.
Z	Function used in the solution to a partially penetrating well.

Symbol	Description
$\{0\}$	Vector of zeroes.
$\{0\}_4$	Vector of zeroes of length 4.

CHAPTER I

INTRODUCTION

Introduction

The modeling of seepage under dams and groundwater flow in aquifers is of significant interest. This becomes even more important in our modern times with increasing interest in the flow of pollutants. The unsolved environmental issues regarding our hazardous and toxic waste problems must be resolved, and significant resources must be placed on this effort. Some military bases are contaminated with hazardous waste that has entered the groundwater domain. A groundwater model that takes into account contaminant flow is therefore critical. The state of the art has advanced in various ways over the years to achieve better and better solutions. However, of unusual occurrence is the application of the tools that engineers in one discipline have developed to problems of other disciplines. What is said is, "We don't do it that way." Because of the author's diverse background, a unique feature of the work in this dissertation is that the tools developed by structural and aerospace engineers are applied to a problem typically addressed by others.

Scope of Dissertation

This dissertation concentrates effort on the complicated, real-world problem of seepage and groundwater flow in three ways:

1. The development and application of new and innovative computational techniques for a more effective solution procedure.

2. The application of techniques and software developed by structural and aerospace engineers to a civil engineering problem (technology transfer).

3. The development and application of a three-dimensional (3-D) seepage package and groundwater model using the latest grid generation and scientific visualization techniques.

More detail will now be given to the different parts.

Two-Dimensional (2-D) Improvements

A 2-D finite element method (FEM) seepage package has been developed by the author which has three parts as follows:

1. Interactive graphics grid generation.

2. Steady-state seepage analysis for both confined and unconfined problems, homogeneous or inhomogeneous media, isotropic or anisotropic soil, plane and axisymmetric flow.

3. Interactive graphics postprocessor.

This package has been distributed worldwide and has advanced the state of the art. Further advancements in automatic flow net generation for a 2-D problem from the perspective of aerospace engineering will first be presented. In fact, the technique of using the Cauchy-Riemann equations to generate a structured orthogonal grid is extended to generate a seepage flow net. Several examples and comparison with other work are also presented.

Three-Dimensional Model

Next, a 3-D seepage package and groundwater model is presented where the primary hardware configuration is the

combination of a Cray YMP and a Silicon Graphics Iris workstation. The individual parts will now be described.

Grid Generation

The program EAGLE (written predominantly by Dr. Joe Thompson, Mississippi State University) has extensive capability in generating structured grids for finite volume flow applications. Many users of FEM programs prefer structured grids (although not required) because of their esthetics and good numerical properties (triangular and tetrahedral elements show bias at times). Therefore, EAGLE was used as the grid generation tool.

Conversion and Data Completion

A module to convert the output from the grid generation program to the FEM seepage format is next described. This includes combining data from different blocks into a single FEM grid, and boundary condition and soil property data must also be supplied. Finally, a bandwidth minimization algorithm must be applied to the grid.

FEM Seepage Analysis

A 3-D version of the 2-D seepage analysis program is next presented. The numerical problems in going from 2-D to 3-D are also described. Like the new 2-D seepage program, the capabilities are:

1. No initial guess of the free surface or any restrictions on the grid shape or numbering in the vicinity of the free surface will be required.

2. Layers with different soil properties are allowed.

3. The program will terminate upon convergence without the user having to specify a specific number of iterations or time steps.

Scientific Visualization

The new 3-D seepage program outputs data compatible with FAST (available from NASA Ames and operational on the Iris Workstation). The techniques to properly format the unconfined flow data are described. Since EAGLE's grids can be plotted with FAST, graphics capability for both preprocessing and scientific visualization are supplied to the user.

Test Problems

The developed 3-D seepage and groundwater model was tested with both theoretically verifiable and real-world problems. These results are presented.

Previous Work

Early attempts at modeling groundwater (Meyer and Kleinecke 1968) assumed horizontal flow in cells of one layer (2.5-D) with a finite difference scheme. However, when the flow became unconfined, the problem became nonlinear, and it was difficult to ensure convergence in a steady-state problem. In fact, one significant aspect of seepage is handling the problem of unconfined flow with a free surface through materials of significantly different characteristics (permeabilities). One example of this is an earth

dam with a relatively impervious clay core with a highly pervious drain installed around it. The rest of the dam is composed of moderately porous material. Figure 1 shows an example with the following soil properties.

Soil	Material	Permeabilities, ft/min k_1	k_2	Angle, deg
1	Rock	$9.3(10^{-2})$	$1.7(10^{-2})$	140
2	Sand	$9.8(10^{-2})$	$2.0(10^{-2})$	0
3	Drain	$9.8(10^{-2})$	$2.0(10^{-2})$	0
4	Shell	$9.8(10^{-1})$	$9.8(10^{-1})$	0
5	Random	$9.8(10^{-3})$	$9.8(10^{-3})$	0
6	Core	$9.2(10^{-5})$	$2.0(10^{-5})$	0
7	Random	$9.8(10^{-3})$	$9.8(10^{-3})$	0
8	Grout	$9.8(10^{-3})$	$2.0(10^{-3})$	140

A finite element analysis (FEA) (Wilson 1969, Zienkiewicz 1971) for 2-D steady-state seepage flow for confined or unconfined flow (Taylor and Brown 1967, Finn 1967) was developed. However, the user was required to orient the elements where the free surface might occur in a special way and give angles along which the free surface should move. This was an early attempt at adaptive mesh methods but did not work well at times. Also, this code could not handle problems that became partially confined and partially unconfined unless the user could pick a priori where the break would be. Some improvement to the convergence problem was made (Neuman and Witherspoon 1970). A different approach to a finite difference solution for

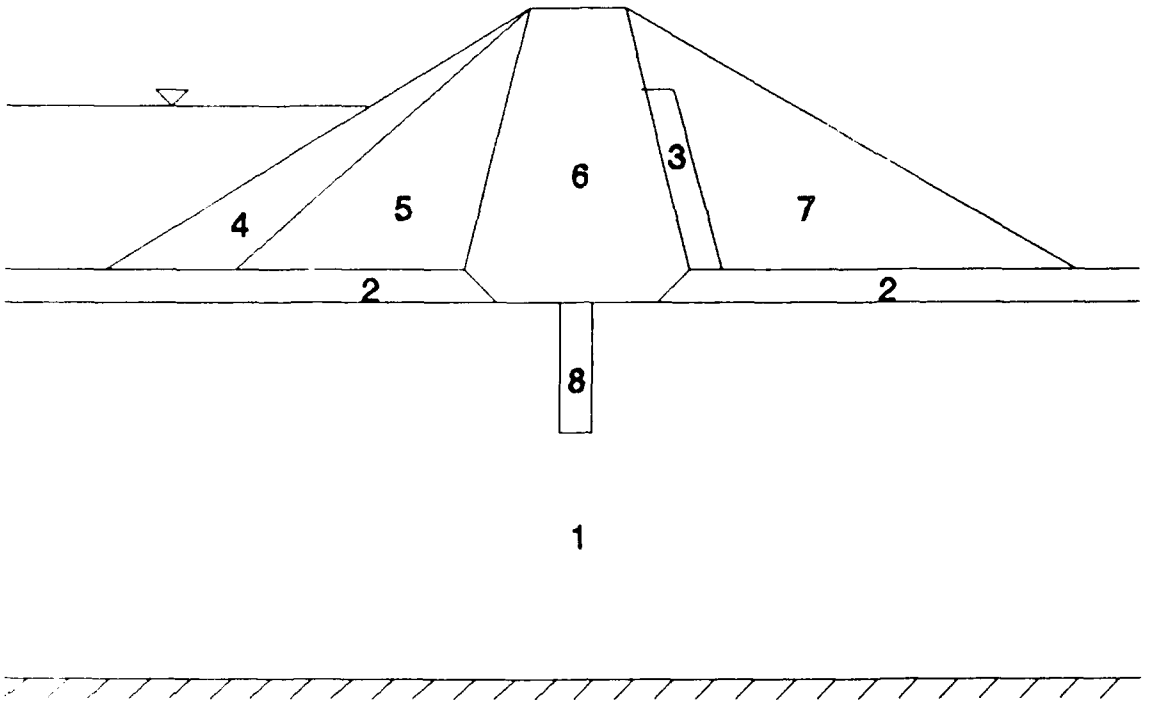


Figure 1. Earth Dam with Different Soil Properties

a transient problem of an initially dry bank (Desai 1970) was taken, but this solution was good for only one boundary condition and only a rectangular mesh. Later, a method of solving the transient seepage problem using the FEM by treating the problem as a series of steady-state problems (France 1971) was developed. Further refinements (Issacs and Mills 1972) to the work of France also were made by modifying elements crossed only by the free surface rather than "adapting" an entire set.

At this time the author developed both 2- and 3-D FEM seepage programs (Tracy 1973a, 1973b). These programs were used in a study of Lock & Dam 26, Alton, IL, with reasonable

results (Hall, Tracy, and Radhakrishnan 1975). However, a subsequent project failed where the grid was produced manually.

It was apparent that grid generation and interactive graphics tools were essential for the successful application of numerical techniques to large real-world problems. Work was then begun on a 2-D FEM grid generator and postprocessor (Tracy 1977a, 1977b, 1977c). The grid generator had two new capabilities as follows:

1. After the user defined points and line segments, the program automatically put them together into subregions. If they were four-sided subregions, a structured algebraic grid was generated.

2. If the subregion had an arbitrary number of sides, a triangular mesh was generated by a simple, yet innovative, technique developed by the author.

The postprocessor could do the following for 2-D FEM output:

1. Numbers.

2. Contours.

3. Vectors.

4. Displaced grid for structures problems.

5. Isometric.

6. Perspective.

As PC's and engineering workstations became more powerful, Apollo and IBM PC versions were created (Tracy 1988). Work was done to plot a 3-D FEM grid (Tracy and Wade 1980), but the work to generate a 3-D grid was left to others. That is, in fact, one of the reasons why the unique capability of

EAGLE (Thompson 1987, Thompson and Gatlin 1988a, 1988b, 1988c) is used in the work documented by this dissertation.

The seepage programs developed at this point have some serious limitations:

1. The scheme to do a steady-state problem by allowing the transient problem to converge with a constant time increment is very slow.

2. When, as shown in Figure 2, the free surface crosses the grid, small and therefore sometimes skewed elements just below the free surface are created.

3. With materials with significantly different permeabilities, a time step good for one layer is totally wrong for the others.

4. An initial guess of the free surface is required.

5. The elements where the free surface would potentially go have to be quadrilaterals numbered in a certain way, destroying one of the major features of the FEM.

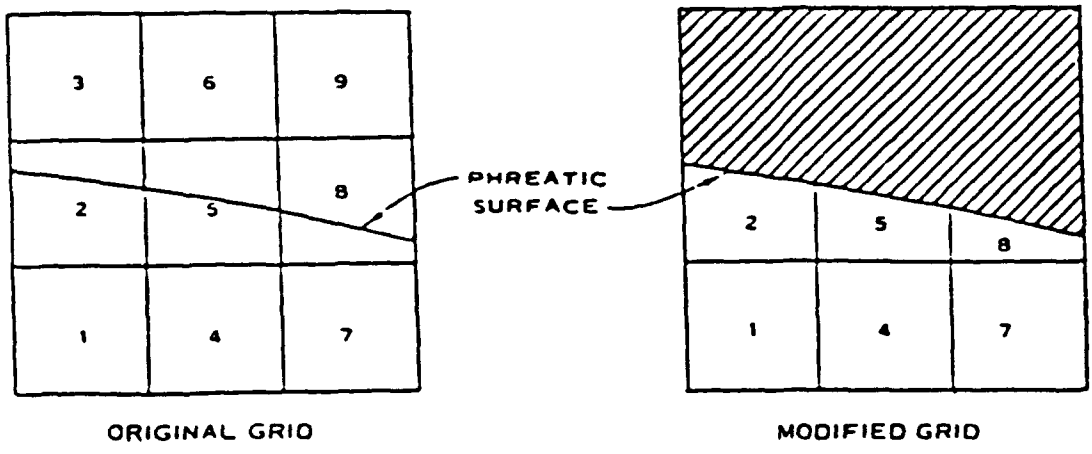


Figure 2. FEM Grid with Free Surface

A new approach where the elements above the free surface are given a very low permeability (Bathe and Khashgoftaar 1979)

was developed which alleviated many of these problems. There is one boundary condition (at the exit point) which was approximated in the work of Bathe and is very difficult to handle. A modified version of Bathe's work with a correct version of exit point boundary conditions (Tracy 1983) was done next. This program was later put into the above-mentioned 2-D seepage package (Biedenharn and Tracy 1987) with the pre- and postprocessor programs. The last improvement is work on the addition of a flow net option (Tracy and Radhakrishnan 1989) which is described in this dissertation with a new aerospace engineering perspective.

CHAPTER II

GOVERNING EQUATIONS

Flow Equations

The equation for unsteady unconfined flow of a compressible fluid in an initially dry compressible porous medium (DeWiest 1966) such as in the earth dam shown in Figure 3 with headwater level $H_H(t)$ and tail water level

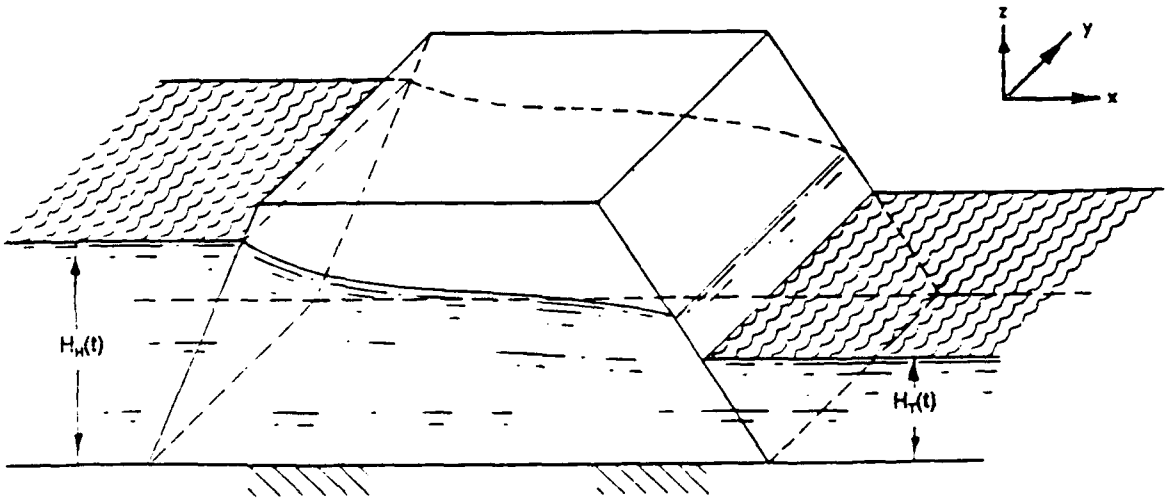


Figure 3. Earth Dam

$H_T(t)$ can be expressed as follows:

$$\frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial(\rho w)}{\partial z} + \frac{\partial(\rho n)}{\partial t} = 0 \qquad (2.1)$$

where

ρ = mass density of the fluid

u, v, w = discharge velocities in the x, y, and z directions, respectively

n = porosity (ratio of the volume of voids to the total volume)

t = time

The actual velocities of the fluid particles are related to the discharge velocities by

$$v_j = \frac{V_j}{n} \qquad j = 1, 2, 3 \tag{2.2}$$

where

v_j = jth component of the actual velocity of the water particles

V_j = jth component of discharge velocity (u, v, or w)

Further, the concentration of the water in the soil-water complex can be measured by a density ϱ defined by

$$\varrho = n\rho \tag{2.3}$$

Equations 2.2 and 2.3 can now be substituted into Equation 2.1 to yield the conservation of mass equation (Anderson, Tannehill, and Pletcher 1984) used by the aerospace engineers as follows:

$$\sum_{j=1}^{3} \frac{\partial(\varrho v_j)}{\partial x_j} + \frac{\partial \varrho}{\partial t} = 0 \tag{2.4}$$

where

x_j = x, y, or z

This equation can be written in vector form by

$$\nabla \cdot (\rho \vec{v}) + \frac{\partial \rho}{\partial t} = 0$$

where $\vec{v}$ is the velocity vector. In like manner, Equation 2.1 can be written

$$\nabla \cdot (\rho \vec{v}) + \frac{\partial (\rho n)}{\partial t} = 0$$

where $\vec{v}$ is the discharge velocity vector. If the soil-water complex is assumed incompressible, ρ is constant. Also, a more general equation can be stated for either confined or unconfined flow. The above two steps yield

$$\nabla \cdot \vec{v} + S_s \frac{\partial h}{\partial t} = 0$$

where

S_s = specific storage

h = total head or potential

Darcy's Law (Maasland 1957) for laminar flow for discharge velocity is given by

$$\vec{v} = - \boldsymbol{K} \cdot \nabla h \qquad (2.5)$$

where **K** is a 3 X 3 permeability tensor. So now the equation for unsteady flow becomes

$$\nabla \cdot (\boldsymbol{K} \cdot \nabla h) = S_s \frac{\partial h}{\partial t} \qquad (2.6)$$

Steady-State Solution

Because this work solves the difficult problem of 3-D unconfined flow with no restriction on the grid, automatic

flow net generation, and building a working and tested tool for use by industry and federal installations, it concentrates on a steady-state solution. If a steady-state solution is approached from solving a nonlinear equation, then the following equation is used.

$$\nabla \cdot (\boldsymbol{K} \cdot \nabla h) = 0 \tag{2.7}$$

Finally, in the special case where a homogeneous, isotropic medium is modeled, the total head satisfies Laplace's Equation,

$$\nabla^2 h = 0 \tag{2.8}$$

Equation 2.8 will be used to generate a flow net for 2-D problems.

CHAPTER III

COMPUTER GENERATED FLOW NETS

Introduction

The graphical construction of flow nets by hand to compute the quantity of flow, exit gradient, etc. is a standard engineering tool of soils engineers. However, these are extremely tedious to construct by hand because equipotential lines and flow lines must be drawn in such a way that curvilinear squares result. One significant aspect of this research effort is that numerical grid generation techniques of aerospace engineers used to generate an orthogonal grid (Thompson, Warsi, and Mastin 1985) can be extended to construct a flow net for various boundary conditions using the Cauchy-Riemann Equations (Crowder and McCuskey 1964). This chapter shows how the FEM has been successfully applied to generate flow nets with emphasis also given to differences in approach from previous work. The major advantage of the techniques described in this chapter is that they improve the quality of the resulting flow nets.

Governing Equations and Basic Approach

The total head or potential ϕ for a homogeneous, isotropic medium for 2-D, steady-state flow (as already shown by Equation 2.8) satisfies Laplace's equation as follows:

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0 \tag{3.1}$$

The stream function ψ also satisfies Laplace's Equation,

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = 0 \tag{3.2}$$

Therefore, a complex potential Φ exists as follows:

$$\Phi = \phi + i\psi$$

Since ϕ and ψ are conjugate harmonic functions, the Cauchy-Riemann equations now hold.

$$\begin{aligned} \frac{\partial \phi}{\partial x} &= \frac{\partial \psi}{\partial y} \quad or \quad \phi_x = \psi_y \\ \frac{\partial \phi}{\partial y} &= -\frac{\partial \psi}{\partial x} \quad or \quad \phi_y = -\psi_x \end{aligned} \tag{3.3}$$

It should be noted here that the stream function is often defined as a velocity-type term. However, in this work it is a gradient-type term. That is, ψ is defined by

$$\psi = \int_c \phi_x dy - \phi_y dx$$

as compared to

$$\psi = \int_c u dy - v dx$$

A property of such functions is that constant lines of ϕ are orthogonal to constant lines of ψ. The flow net consists of ϕ = constant lines and ψ = constant lines constructed in such a way that the resulting picture consists of curvilinear squares. The concept of automatically generating the flow net is fairly straightforward and involves the following three steps:

1. Perform a normal FEM solution determining the total head h (same as potential ϕ) at each node and the quantity of flow, Q, passing through the system. Also compute the shape factor, f, from

$$Q = k(h_u - h_d) f \qquad (3.4)$$

where h_u is the upstream head, h_d is the downstream head, k is the permeability, Q is the flow, and f is the shape factor.

2. Determine the boundary conditions for the stream function and perform a second FEM solution to obtain values of ψ at each node.

3. Contour the two sets of data to construct the flow net. The intervals for each are determined using the shape factor which, by definition, is

$$f = \frac{N_f}{N_e}$$

where N_e is the number of equipotential drops, and N_f is the number of flow paths.

Earlier work (Christian 1980a, 1980b, 1983; Aalto 1984; Christian 1987) determined the boundary values for the stream function solution (step 2) numerically, whereas in this work a more fundamental technique is used. Here, the

Cauchy-Riemann equations are used to determine the correct boundary conditions.

Modified Cauchy-Riemann Conditions

A modified form of Equation 3.3 will be applied to determine the boundary conditions for the stream-function computation (step 2). On the boundary, a tangent-normal (T-N) coordinate system is used as illustrated in Figure 4. It is created by first constructing a parallel coordinate system x' - y' at the

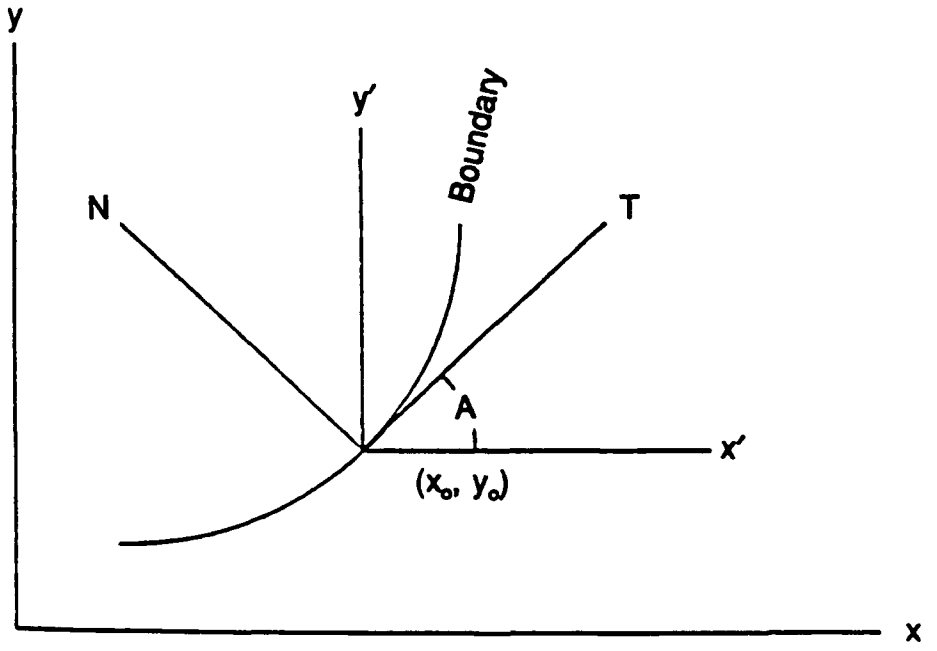


Figure 4. Tangent-Normal System

boundary point (x_o, y_o) and then rotating the prime system an angle A. The local and global coordinate systems are related by:

$$x' = x - x_o$$

$$y' = y - y_o$$

and the tangent-normal coordinate system is related to the x' - y' system by:

$$\begin{pmatrix} T \\ N \end{pmatrix} = \begin{pmatrix} \cos A & \sin A \\ -\sin A & \cos A \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}$$

where A is the angle between the two axes. Therefore,

$$\begin{pmatrix} \phi_T \\ \phi_N \end{pmatrix} = \begin{pmatrix} \cos A & \sin A \\ -\sin A & \cos A \end{pmatrix} \begin{pmatrix} \phi_x \\ \phi_y \end{pmatrix} \tag{3.4}$$

$$\begin{pmatrix} \psi_T \\ \psi_N \end{pmatrix} = \begin{pmatrix} \cos A & \sin A \\ -\sin A & \cos A \end{pmatrix} \begin{pmatrix} \psi_x \\ \psi_y \end{pmatrix} \tag{3.5}$$

where

$$\phi_T = \frac{\partial \phi}{\partial T} \qquad \psi_T = \frac{\partial \psi}{\partial T}$$

$$\phi_N = \frac{\partial \phi}{\partial N} \qquad \psi_N = \frac{\partial \psi}{\partial N}$$

Applying Equation 3.3 to Equation 3.5 yields

$$\begin{pmatrix} \psi_T \\ \psi_N \end{pmatrix} = \begin{pmatrix} \cos A & \sin A \\ -\sin A & \cos A \end{pmatrix} \begin{pmatrix} -\phi_y \\ \phi_x \end{pmatrix}$$

Collecting terms gives

$$\begin{pmatrix} \psi_N \\ -\psi_T \end{pmatrix} = \begin{pmatrix} \cos A & \sin A \\ -\sin A & \cos A \end{pmatrix} \begin{pmatrix} \phi_x \\ \phi_y \end{pmatrix} \tag{3.6}$$

Since the right-hand side of Equations 3.4 and 3.6 are the same, the left-hand sides can be equated to obtain

$$\begin{aligned} \phi_T &= \psi_N \\ \phi_N &= -\psi_T \end{aligned} \tag{3.7}$$

Equation 3.7 can now be used to solve for the boundary conditions for the stream function.

Examples

Several examples will now be presented to illustrate the procedure.

Four-Sided Regions

Two problems showing examples of the basic four-sided region are now given. They are flow under a weir and two versions of flow in a partially penetrating slot.

Confined Flow under a Weir

This simple problem (Figure 5) clearly demonstrates the concepts presented in this chapter. Note that there are four boundaries, with constant head specified on two lines (segments AFE and BCD) and a no-flow (impervious) condition specified on the other two lines (segments AB and ED).

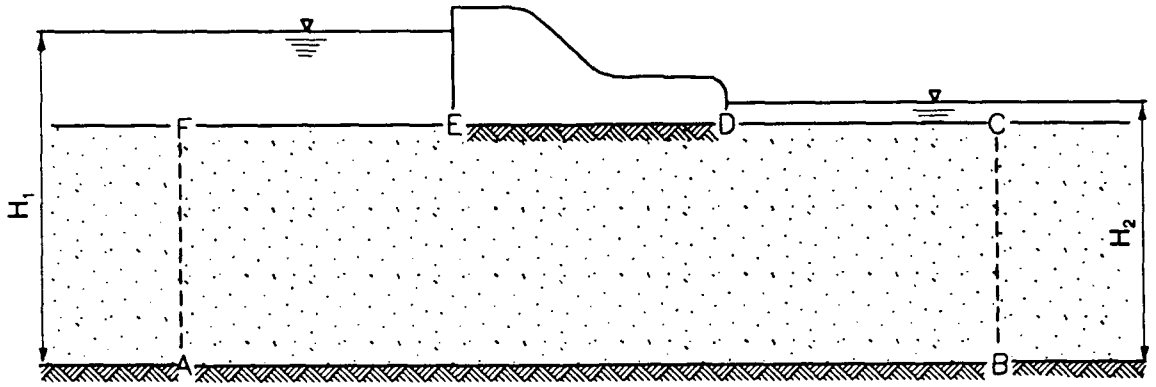


Figure 5. Weir Problem

On the $\phi = H_1$ and $\phi = H_2$ boundaries (AFE and BCD)

$$\phi_T = 0$$

since H_1 and H_2 are constants. But from Equation 3.7 we get

$$\phi_T = \psi_N$$

So the new boundary condition is

$$\psi_N = 0$$

On the impervious boundaries no flow enters, so the normal component of velocity, v_N, is zero. Thus, using Darcy's Law for a homogeneous, isotropic medium,

$$v_N = - k\phi_N = 0$$

it follows that

$$\phi_N = 0$$

Applying Equation 3.7 to the above equation yields

$$\Phi_N = -\psi_T = 0$$

or

$$\psi_T = 0 \qquad (3.8)$$

Now

$$d\psi = \psi_T \, dT + \psi_N \, dN \qquad (3.9)$$

Substituting Equation 3.8 into the above equation and noting that dN = 0 on the boundary gives

$$d\psi = 0$$

or

$$\psi = constant \qquad (3.10)$$

The total amount of stream function can be shown to be

$$\psi_{total} = \frac{Q}{k} = f(h_u - h_d)$$

Thus for the impervious boundaries (using Equation 3.10), set

$$\psi = \psi_1 = 0 \qquad on\ AB$$

$$\psi = \psi_2 = \psi_{total} \qquad on\ ED$$

(In the actual computer program, a constant is added to ψ_1 and ψ_2 so their values will be higher than the maximum y value of the grid to keep the FE program from solving an unconfined flow problem.)

Notice that the boundary conditions are exactly reversed. The impervious boundary in the first problem is

replaced as a constant head boundary, and a constant head boundary has been replaced as a no-flow or impervious boundary.

A wide variety of problems involving four-sided regions can be handled by the concept demonstrated here. The first example is given in Figure 6 where a weir similar to the one of Figure 5, but with sheet piles added, is shown (BC = AJ = 40 ft, DE = GH = 10 ft, CD = IJ = 30 ft, and FG = 40 ft). Figure 7 shows the computer generated flow net for this problem. The sheet piles remain part of the one continuous impervious boundary (DEFGHI in Figure 6). Points D and F have the same (x, y) coordinates as do points G and I.

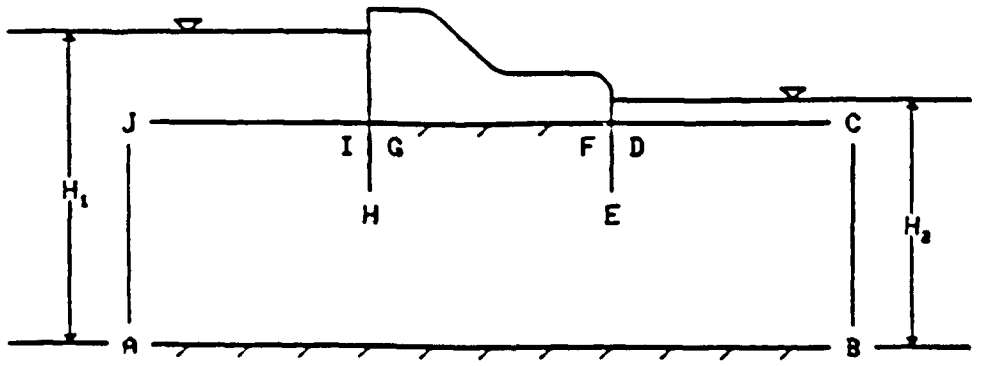


Figure 6. Weir Problem with Sheet Piles

Partially Penetrating Slot

The second example is confined flow through a partially penetrating slot. Both the case of specified head and specified constant discharge velocity at the slot are considered.

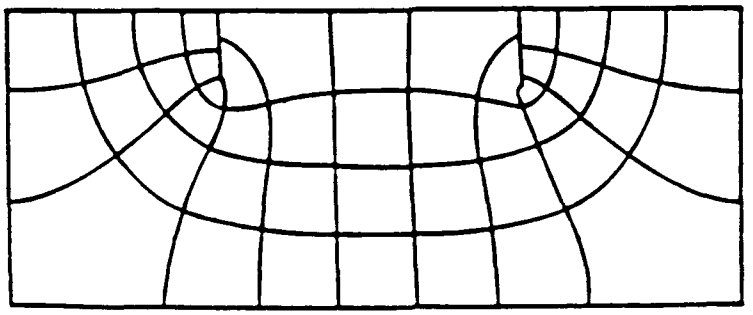


Figure 7. Flow Net for Weir Problem

Head specified at the slot

Figure 8 shows that head or potential is specified at two boundaries (at the headwater level on BC and at the slot along DEF), and the other two boundaries are impervious (along the center line and base FAB and the top CD). Since this problem is topologically identical to the weir problem, no further work is needed to solve it.

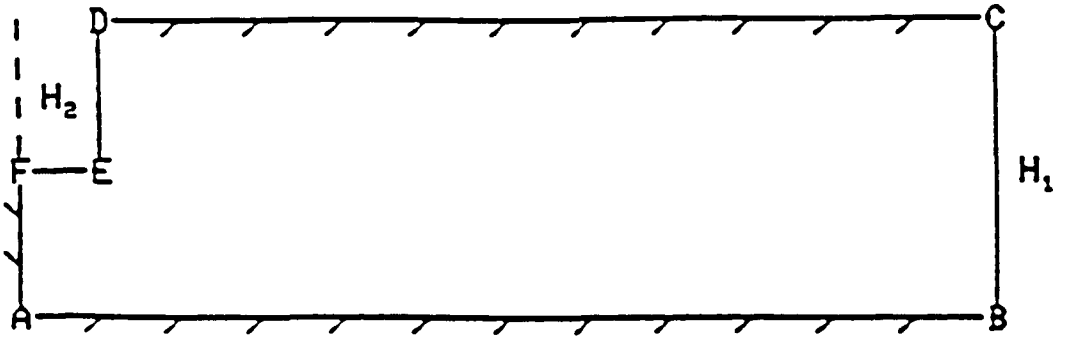


Figure 8. Partially Penetrating Slot

Constant discharge velocity
specified at the slot

This version of the slot problem consists of constant discharge velocity specified on one of the four boundaries, a constant head on one boundary, and impervious conditions

on the other two (Figure 9). Line segments EAB and DC are

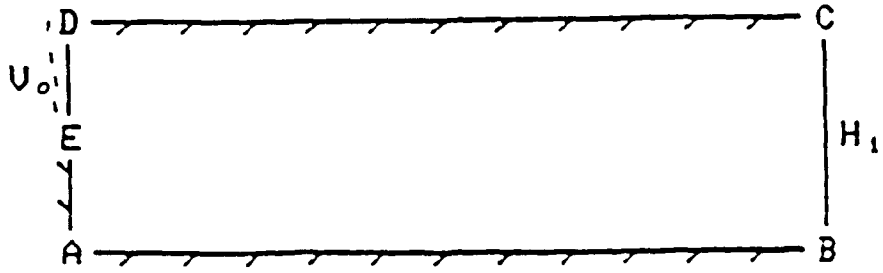


Figure 9. Specified Discharge Velocity

impervious, line segment BC has constant head H_1 specified, and line segment DE has a specified discharge velocity, V_o. On the $\phi = H_1$ boundary, use as before,

$$\psi_N = 0$$

On the boundary DE,

$$v_N = - k\phi_N = v_o$$

or

$$\phi_N = - \frac{v_o}{k}$$

Equation 3.7 can again be applied to obtain

$$\phi_N = -\psi_T = -\frac{v_o}{k}$$

so

$$\psi_T = \frac{v_o}{k}$$

Starting again with Equation 3.9,

$$d\psi = \psi_T \, dT + \psi_N \, dN$$

and observing that dN = 0 on the boundary again gives

$$d\psi = \frac{v_o}{k}\, dT$$

Integrating,

$$\psi = \frac{v_o}{k} T + c$$

where c is a constant to be evaluated. In this particular problem y coincides with T along DE, so

$$\psi = \frac{v_o}{k} y + c$$

Let

$$\psi = \psi_E \quad at \quad y = y_E$$

then

$$c = \psi_E - \frac{v_o}{k} y_E$$

and

$$\psi = \frac{v_o}{k} (y - y_E) + \psi_E$$

Now, on the impervious boundaries, apply the ψ = constant boundary condition as follows:

$$\psi = \psi_E \quad on\ EAB$$
$$\psi = \psi_D \quad on\ CD$$

where ψ_D is computed by

$$\psi_D = \frac{v_o}{k}(y_D - y_E) + \psi_E$$

The boundary conditions for the stream-function solution are now completely defined.

Dupuit's Problem

Dupuit's problem deals with unconfined flow in an earth dam with vertical sides (Figure 10). Line segment AB is impervious, line segments AF and BC have constant head specified, and line segment CD has the boundary condition

$$\phi = y$$

The position of the free surface (FD) must be determined from the FEM solution. Once determined, line segment FD becomes a flow line and is treated exactly like an impervious boundary. Also, the region above the free surface

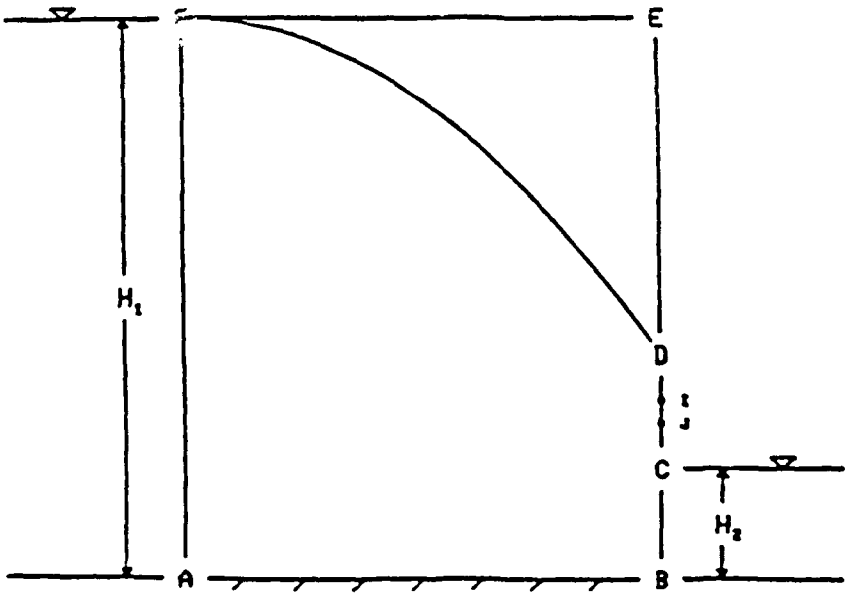


Figure 10. Dupuit's Problem

(triangular region FDE) is not used for the second solution. Rather, a new grid with the phreatic surface being a new boundary is used. We will now determine the ψ boundary conditions.

On the $\phi = H_1$ and $\phi = H_2$ boundaries use, as before,

$$\psi_N = 0$$

On the impervious boundaries apply the ψ = constant boundary condition as follows:

$$\psi = \psi_1 \qquad \text{on } AB$$

$$\psi = \psi_2 = \psi_1 + \psi_{total} \qquad \text{on } FD$$

On the boundary CD,

$$\phi = y$$

This, however, is insufficient information to determine the new boundary conditions. Therefore, the normal component of discharge velocity v_N is first computed for each node on the boundary CD. It is assumed that points C and D are node points, and there are typically intermediate node points such as nodes I and J in Figure 10 as well. Then for each node,

$$v_N = -k\phi_N$$

$$\phi_N = -\psi_T$$

$$\psi_T = \frac{v_N}{k}$$

It is further assumed that v_N (and therefore ψ_T) varies linearly between node points. So between two node points I and J having ψ values of ψ_I and ψ_J and ψ_T values of ψ_{TI} and ψ_{TJ}, ψ_T is approximated as

$$\psi_T = \frac{(\psi_{TJ} - \psi_{TI})}{T_J - T_I} (T - T_I) + \psi_{TI}$$

Integrating as before gives

$$\psi = \frac{1}{2} \frac{(\psi_{TJ} - \psi_{TI})}{T_J - T_I} (T - T_I)^2 + \psi_{TI}(T - T_I) + \psi_I$$

Solving for ψ_J gives

$$\psi_J = \frac{1}{2} (\psi_{TI} + \psi_{TJ}) (T_J - T_I) + \psi_I$$

The first set of node points starts with point I corresponding to point D (Figure 10). Thus,

$$\psi_I = \psi_D$$

Also, using

$$\psi_T = \frac{v_N}{k} = 0$$

(since V_N is zero all along the flow line FD), begin with

$$\psi_{TI} = 0$$

Point D, being the exit point, causes significant numerical problems (to be discussed in Chapter IV). With this start and the values of V_N computed for all the remaining nodes on

the surface of seepage from the FEM solution, the nodes along DC can now be processed consecutively until the tailwater point C is processed. The boundary conditions are now fully determined for the stream-function calculation. Note that the only restriction on ψ_1 is that it be large enough to prevent the FEM analysis program from solving an unconfined flow problem while solving for the stream function.

The above formulation is not restricted to Dupuit's Problem but can also be applied to a wide variety of quadrilateral-type earth dams. The only restriction is that the problem should have the five basic boundaries of impervious base, specified headwater and tailwater, free surface, and surface of seepage. Figure 11 shows the results for Dupuit's Problem where by Figure 10, AB = AF = 100 ft, and BC = 20 ft. Figure 12 shows the computer generated flow net for an earth dam.

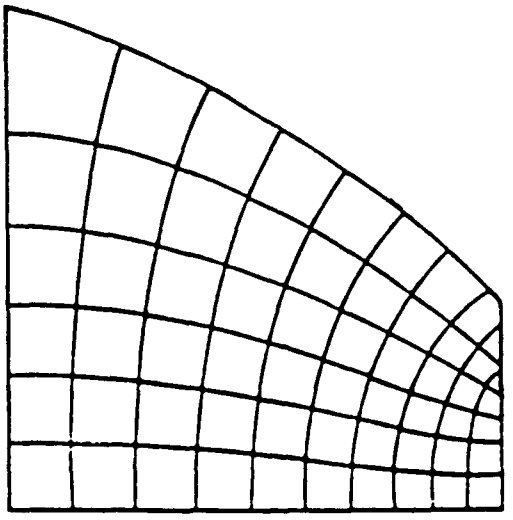


Figure 11. Results for Dupuit's Problem

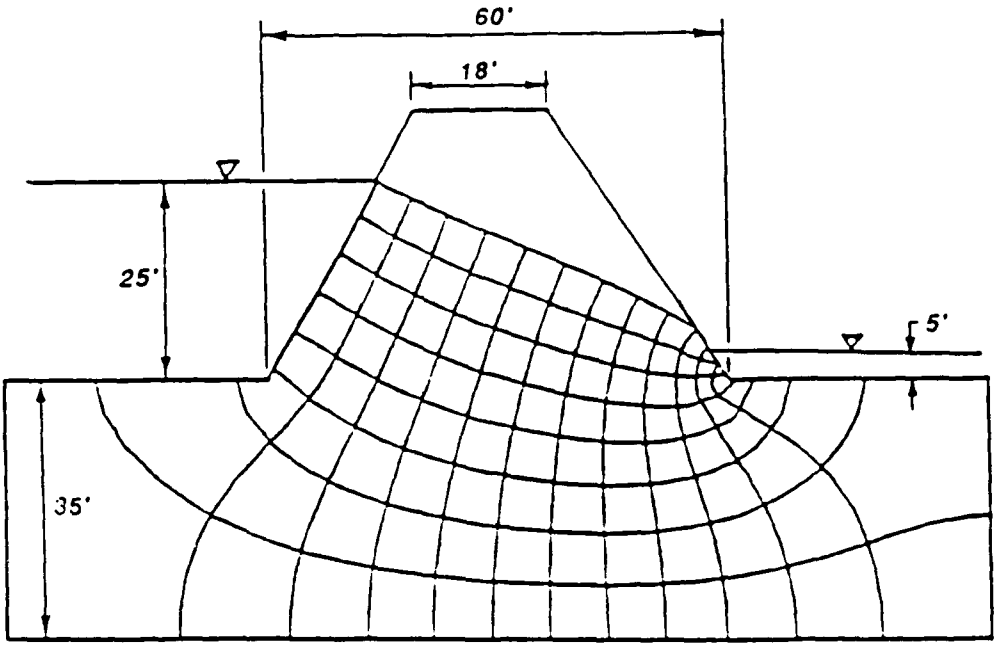


Figure 12. Earth Dam Problem

Anisotropic Soil

Flow nets for an anisotropic soil can be produced equally well for one layer with this technique. The number of flow lines and equipotential lines is computed the same way as with the isotropic case. However, the resulting plot will now have curvilinear quadrilaterals instead of curvilinear squares. To get curvilinear squares, the geometry and permeability must be transformed in the traditional way as follows:

$$x' = x$$

$$y' = \sqrt{\frac{k_H}{k_V}}\, y$$

$$k' = \sqrt{k_H k_V}$$

where k_H and k_V are the horizontal and vertical permeabilities, respectively. If the principal permeabilities are not coincident with the x-y axis, x and y must represent a rotated coordinate system in the equations, except that now k_H and k_V are more properly rendered k_1 and k_2.

Figure 13 shows an anisotropic problem consisting of a weir

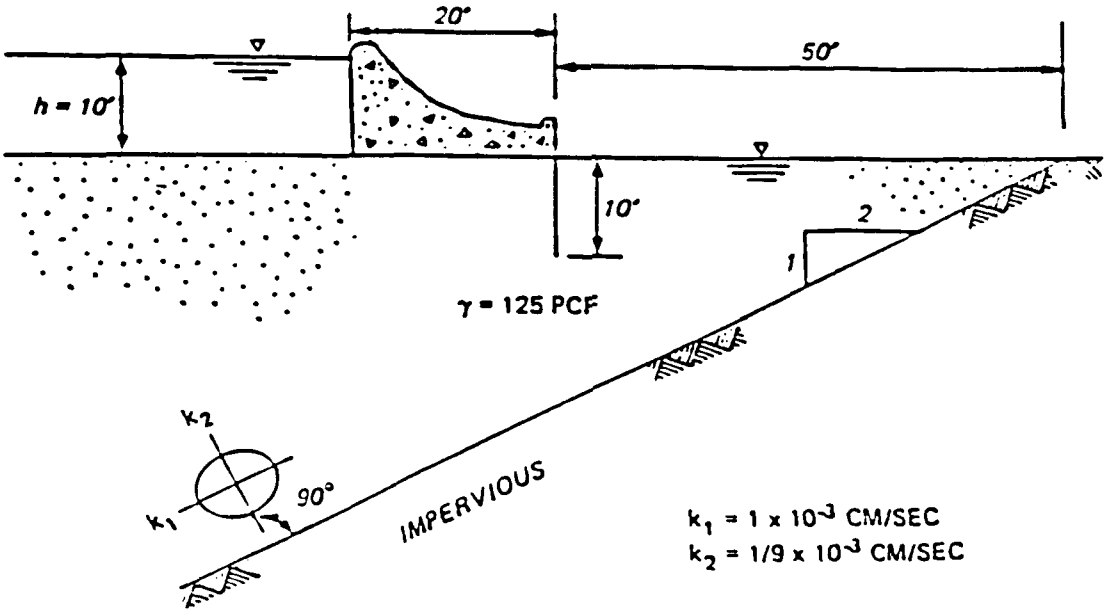


Figure 13. Weir on Anisotropic Soil

with a sheet pile and a sloping impervious bottom. Note that the principal permeabilities are not parallel to the x-y axis. Figure 14 shows the flow net if the soil is assumed isotropic, and Figure 15 shows the results for the anisotropic case.

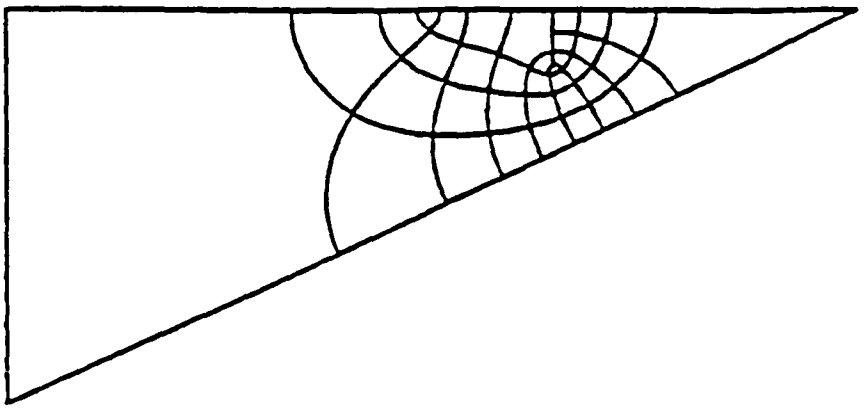


Figure 14. Isotropic Results

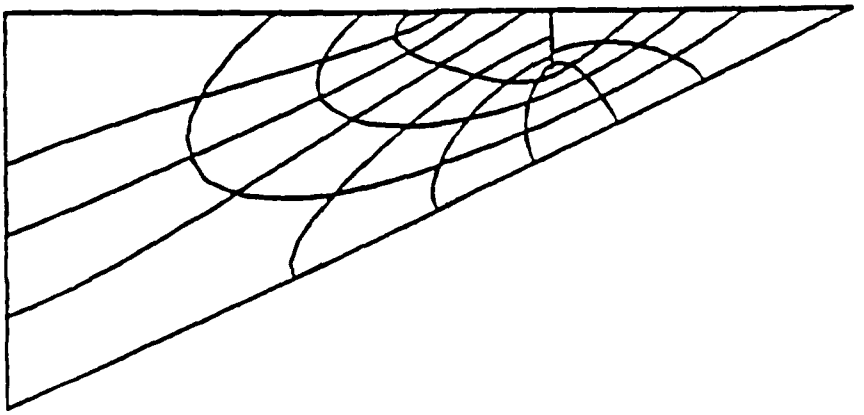


Figure 15. Anisotropic Results

Multilayered Problems

Multilayered problems where all the flow lines originate or end in the same material type can also be solved. To understand how, first consider the two-layered system shown in Figure 16. The procedure is done the same as the single-layered case except that in the second solution for flow lines the permeabilities are modified. The reason is that when an equipotential or flow line crosses a boundary between materials of different permeabilities, it is bent.

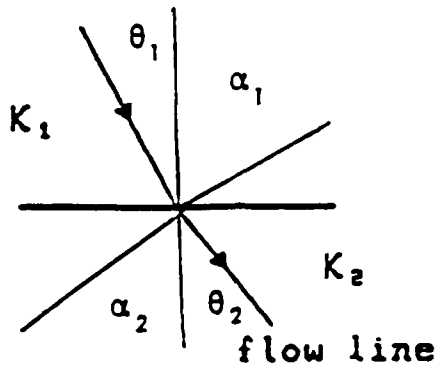


Figure 16. Flow across Boundary

However, since both regions are isotropic, the flow lines must remain perpendicular to the equipotential lines on both sides of the boundary. Flow lines are bent as follows (Freeze and Cherry 1979) (Figure 16):

$$\frac{\tan \theta_1}{\tan \theta_2} = \frac{k_1}{k_2} \qquad (3.11)$$

In a similar manner, since equipotential lines are perpendicular to flow lines, equipotential lines are bent as follows:

$$\frac{\tan (\frac{\pi}{2} - \alpha_1)}{\tan (\frac{\pi}{2} - \alpha_2)} = \frac{k_1}{k_2}$$

or

$$\frac{\cot \alpha_1}{\cot \alpha_2} = \frac{k_1}{k_2}$$

Finally, taking the reciprocal of both sides gives

$$\frac{\tan \alpha_1}{\tan \alpha_2} = \frac{k_2}{k_1} \qquad (3.12)$$

When the second FEM solution is completed, the program treats the flow lines as equipotential lines. Therefore, the flow lines are bent according to Equation 3.12 instead of Equation 3.11. To compensate for this, the ratios of permeability must be reversed. This can be accomplished, for instance, by reversing k_1 and k_2. For a multilayered problem, however, it is best to replace the k's with their reciprocals, respectively. Thus, the ratio of k's between material types 1 and 2 becomes

$$\frac{k_2'}{k_1'} = \frac{\left(\frac{1}{k_2}\right)}{\left(\frac{1}{k_1}\right)} = \frac{k_1}{k_2}$$

Between materials 2 and 3 it becomes

$$\frac{k_3'}{k_2'} = \frac{\left(\frac{1}{k_3}\right)}{\left(\frac{1}{k_2}\right)} = \frac{k_2}{k_3}$$

This enables the procedure to work for any number of layers.

Figure 17 shows a three-layered problem with the dividing line between the layers being equidistant, and Figure 18 shows the computer generated results. Since the problem has isotropic soil layers, the equipotentials and flow lines remain perpendicular to each other. However, in one layer

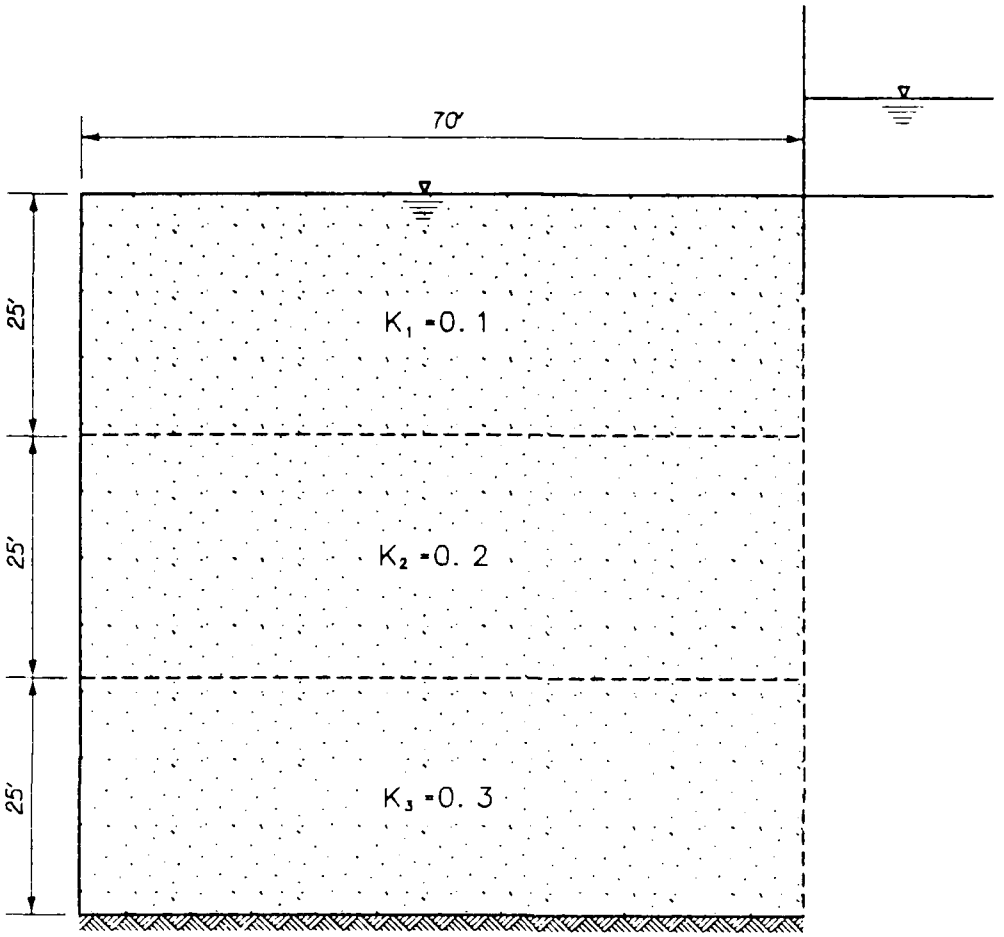


Figure 17. Three-Layered Problem

there will remain curvilinear squares, and in the other two layers there will be curvilinear rectangles.

Axisymmetric Case

The flow net as traditionally defined (a plot of equal potential drops and equal flow paths constructed in such a way as to produce curvilinear squares) does not exist for an axisymmetric problem (Figure 19 shows the result for a fully penetrating well which has a well radius of 1 ft, a radius of influence of 51 ft, a depth of 20 ft, and a permeability

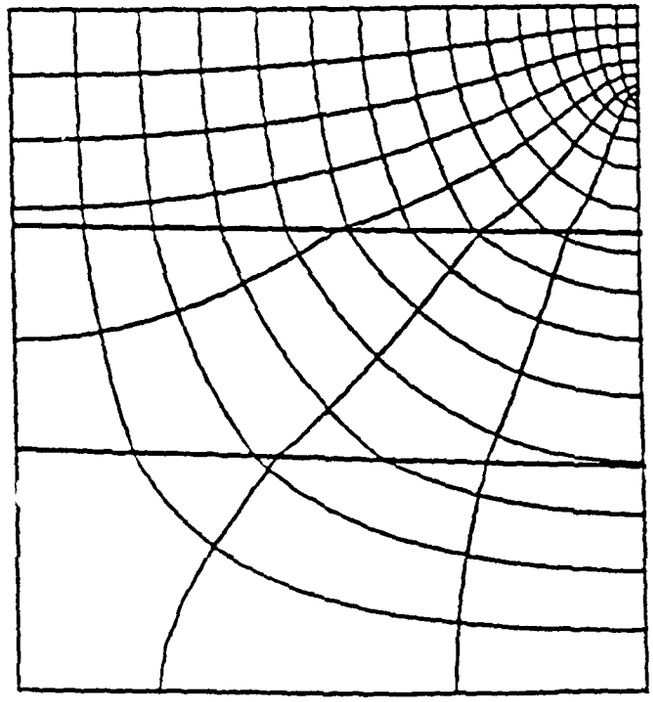


Figure 18. Results for Three-Layered Problem

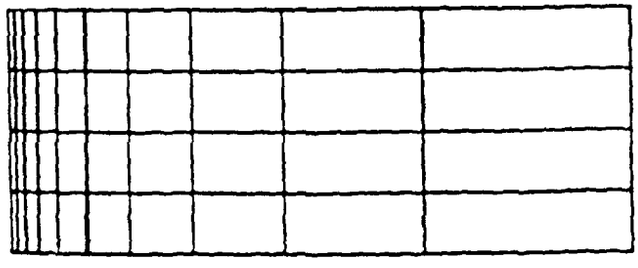


Figure 19. Fully Penetrating Well "Flow Net"

of 0.1 ft/min.). Notice that tall, thin rectangles become short and fat proceeding from the well outward. No addition or subtraction of flow lines can change this situation. The technique outlined in this chapter can be used to produce a plot showing mutually perpendicular equipotential and flow lines, but the number of each is arbitrary.

Comparison with Other Work

Two problems presented by Christian were dealt with using the techniques presented in this chapter and the results compared. The first problem is confined flow in an anisotropic medium as shown in Figure 20. Figure 20 also shows Christian's results with the author's results in Figure 21. Note that the results are the same. A more difficult problem is unconfined flow through a two-zoned earth dam with $k_B = 2k_A$. The problem and Christian's results are shown in Figure 22, and the author's results are shown in Figure 23. With curvilinear squares in Region A, curvilinear rectangles of size 2 to 1 should occur in Region B. Note that the results by the author are much closer to this result. Also, due to numerical complexities at the exit point that the author avoids, an extra wiggle occurs in Christian's work near the exit point. Clearly, the more fundamental approach generates a superior product.

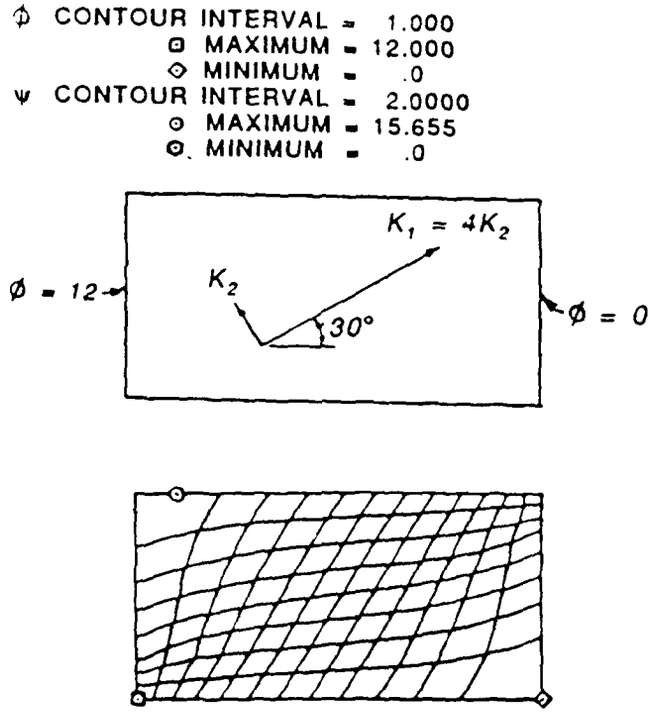


Figure 20. Confined Flow in an Anisotropic Medium

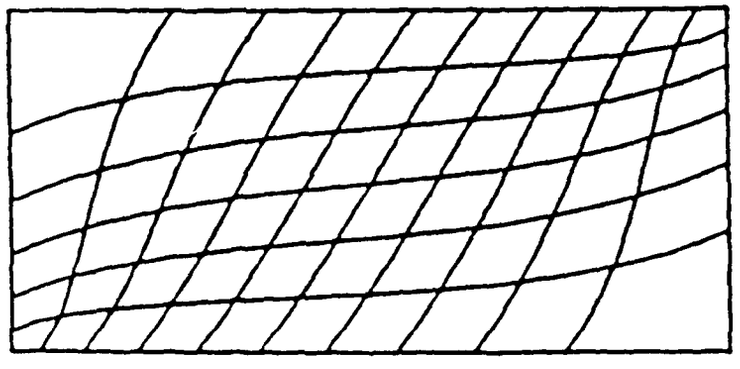


Figure 21. Author's Results

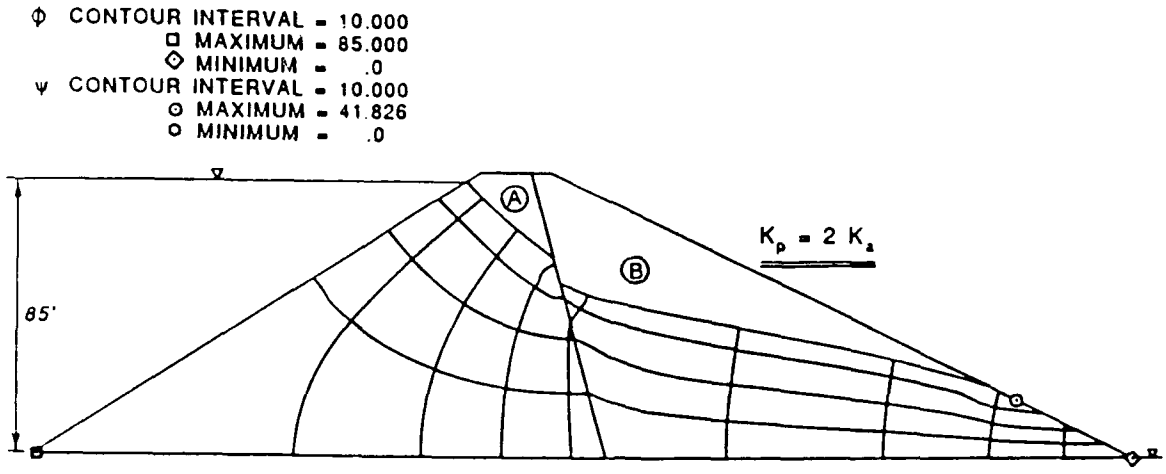


Figure 22. Unconfined Flow through Two-Zoned Dam

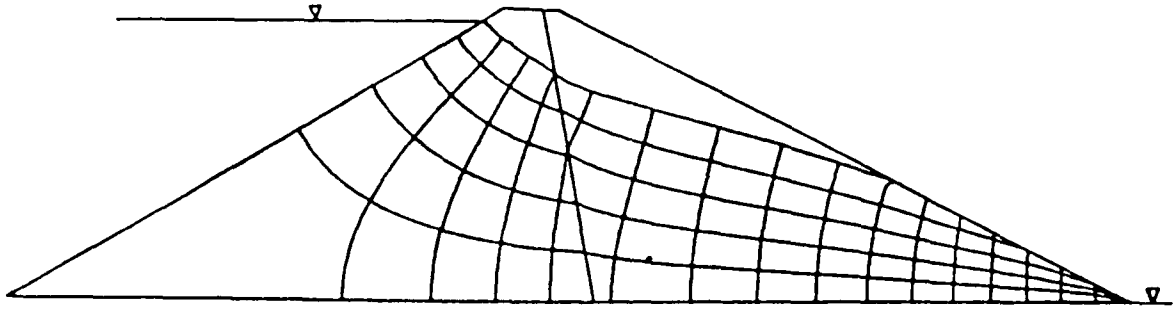


Figure 23. Author's Results

CHAPTER IV
THREE-DIMENSIONAL NUMERICS

Introduction

The goal of this work is to allow a completely general 3-D FE grid for both confined and unconfined flow problems. Further, the user is not to be required to make an initial guess of free surface areas or be concerned about the number of iterations required for convergence. The advantage of this approach is that a totally structured grid, partially structured and partially unstructured grid, or totally unstructured grid can be used. However, going from 2- to 3-D is significant. For example, in a 2-D unconfined flow problem there is typically one exit point to cause computational problems. In a 3-D problem there is an exit line instead of an exit point, and there can be any number of them. One example is an aquifer or cofferdam with several wells. The numerics invo'ved in the 3-D solution will now be discussed.

Finite Volume versus Finite Element Comparison

An unusual approach will be used to introduce the FE formulation. Typically, a variational approach is used to derive the governing FEM equations. This works very well

and is at times physically based such as in structures problems where the principle of virtual work in variational form can be used. What will be done here, however, is to first compare finite volume and FEM by considering one-dimensional (1-D), steady-state seepage flow in the region shown in Figure 24 with three finite volume cells having dimensions of Δx by Δy by 1.

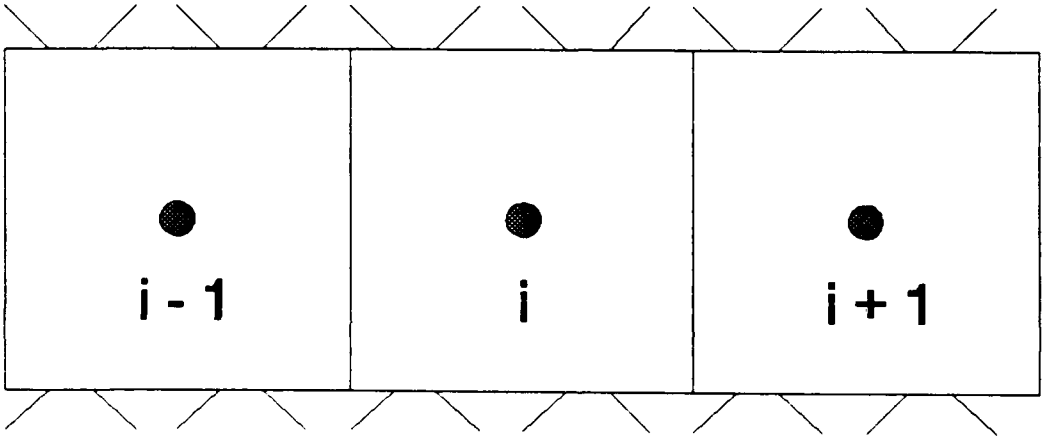


Figure 24. One-Dimensional Seepage Flow

First, putting Equation 2.5 back into Equation 2.7 gives

$$\nabla \cdot \vec{v} = 0 \qquad (4.1)$$

Integrating finite volume i over its volume gives

$$\int_V \nabla \cdot \vec{v} \, dV = 0$$

Using the Divergence Theorem produces

$$\int_S \vec{v} \cdot d\vec{S} = 0$$

Since flow is horizontal, only the two vertical boundaries have flux. So the equation can be approximated by

$$u_{i+\frac{1}{2}} \Delta y - u_{i-\frac{1}{2}} \Delta y = 0 \qquad (4.2)$$

The 1-D version of Equation 2.5 for homogeneous, isotropic flow is

$$u = - k \frac{dh}{dx}$$

Using a linear version of Richardson's Interpolation and gathering information on either side of the boundary produces the standard approximation at the boundaries of

$$u_{i+\frac{1}{2}} = \frac{h_{i+1} - h_i}{\Delta x}$$

$$u_{i-\frac{1}{2}} = \frac{h_i - h_{i-1}}{\Delta x}$$

The above series of equations can be substituted in Equation 4.2 to obtain

$$- k \frac{h_{i+1} - h_i}{\Delta x} \Delta y + k \frac{h_i - h_{i-1}}{\Delta x} \Delta y = 0$$

When terms are collected, one obtains

$$- \frac{k \Delta y}{\Delta x} (h_{i+1} - 2h_i + h_{i-1}) = 0 \qquad (4.3)$$

The term in parenthesis is the familiar central difference formulation for the second derivative in computational space.

Now the same problem can be considered by using the direct stiffness approach of the FEM (Figure 25). The three cells are now three finite elements with eight nodes. Since the flow is horizontal, nodes 1 and 5, 2 and 6, 3 and 7, and 4 and 8, respectively, have the same head. Therefore, nodes 5 through 8 are considered inactive. A structures

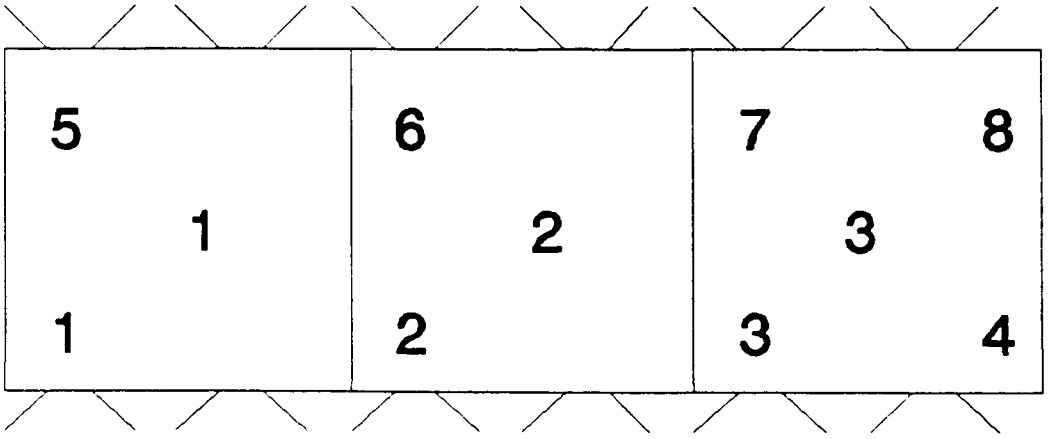


Figure 25. FEM Version

problem will relate forces and displacements by use of Hook's Law via a stiffness matrix as follows:

$$\{F\} = [K]\{u\}$$

where

{F} = nodal forces

[K] = stiffness matrix

{u} = are the nodal displacements

In an analogous manner, in seepage the nodal flows {Q} are linked to nodal heads as follows:

$$\{Q\} = [K]\{h\}$$

Let us now look at element 1. x varies inside the element as follows:

$$x = \frac{1 - \xi}{2} x_1 + \frac{1 + \xi}{2} x_2$$

$$= N_1 x_1 + N_2 x_2$$

where

$$-1 \leq \xi \leq 1$$

The vector form is

$$x = \{N\}^T \{x\}_e \qquad (4.4)$$

where $\{N\}$ is a vector of shape functions and $\{x\}_e$ are the nodal x coordinates for an element. Since nodes 1 and 2 are the only active nodes for element 1, let h vary the same as x in Equation 4.4. Therefore,

$$h = \{N\}^T \{h\}_e \qquad (4.5)$$

where $\{h\}_e$ are the nodal heads for an element. Thus, the isoparametric element formulation (Cook 1981) has been used.

The nodal flows are simply the flux (volume of water per unit time) crossing at that node, so in this case,

$$Q = u\Delta y$$

But

$$u = -ki$$

where the gradient i is in this case

$$i = \frac{dh}{dx}$$

So

$$i = \frac{d\{N\}^T}{dx}\{h\}_e$$

$$= \frac{d\xi}{dx}\frac{d\{N\}^T}{d\xi}\{h\}_e$$

From the definitions,

$$[P] = \frac{d\{N\}^T}{d\xi}$$

$$J = \frac{dx}{d\xi}$$

we get

$$i = J^{-1}[P]\{h\}_e$$

Our goal here is to obtain a matrix [B] so that

$$i = [B]\{h\}_e \qquad (4.6)$$

We see from these equations that

$$[B] = J^{-1}[P] \qquad (4.7)$$

In our 1-D example

$$J = \frac{1}{2}(x_2 - x_1) = \frac{\Delta x}{2}$$

$$[P] = \frac{1}{2}[-1 \quad 1]$$

$$[B] = \frac{1}{\Delta x}[-1 \quad 1]$$

The discharge velocity can now be computed as

$$u = -k[B]\{h\}_e \tag{4.8}$$

which becomes a simple constant in the 1-D problem as follows:

$$u = \frac{k}{\Delta x}(h_1 - h_2)$$

Our goal is to determine the element stiffness matrix $[K]_e$ relating the nodal Q's to the nodal h's. The sign convention used for flow entering the element is positive and for flow exiting the element is negative. Thus,

$$Q_1 = \frac{k\Delta y}{\Delta x}(h_1 - h_2)$$

$$Q_2 = \frac{k\Delta y}{\Delta x}(h_2 - h_1)$$

or

$$\begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix} = \frac{k\Delta y}{\Delta x}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} h_1 \\ h_2 \end{pmatrix}$$

So from the definition

$$\{Q\}_e = [K]_e\{h\}_e \tag{4.9}$$

we get

$$[K]_e = \frac{k\Delta y}{\Delta x}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

This can be written

$$[K]_e = \frac{1}{\Delta x}\begin{pmatrix} -1 \\ 1 \end{pmatrix} [k] \frac{1}{\Delta x}(-1 \quad 1) \Delta x \Delta y$$

The above equation can also be written

$$[K]_e = [B]^T[k][B]\ \Delta x \Delta y$$

So our final goal is reached, which is to have an intuitive basis for the general expression of the element stiffness matrix, which is

$$[K]_e = \int_V [B]^T[k][B]\ dV \qquad (4.10)$$

where now [k] is the matrix version of the general permeability tensor.

General Finite Element Method Formulation

We are now ready to discuss the general FEM equations and techniques for the seepage problem.

Element Formulation

The different aspects of the individual element formulation will now be discussed.

Element Shape

To conform to the output from EAGLE, as well as add a quadratic variation in the interpolation function, a nine-node element is used. As shown in Figure 26, this becomes an eight-node "brick" element with an internal node having coordinate values

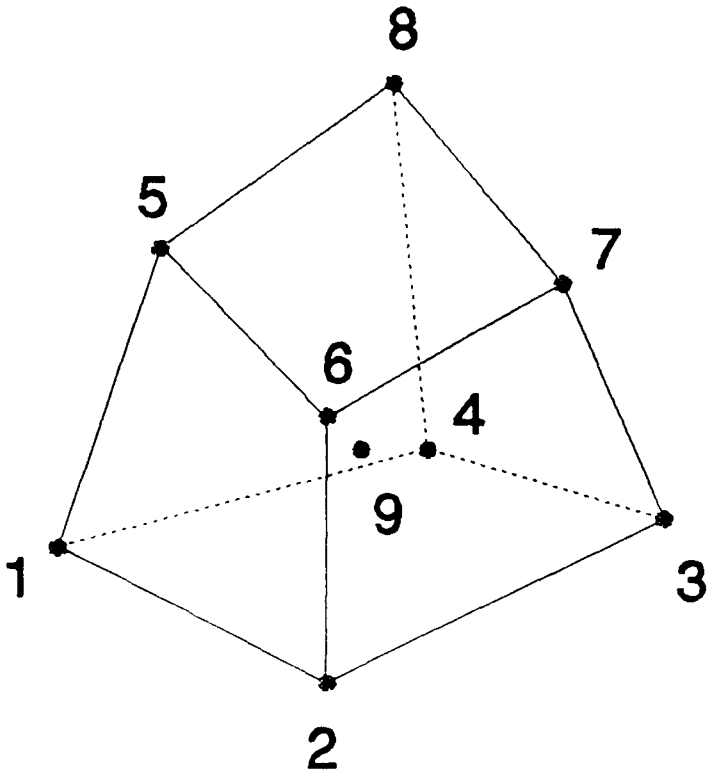


Figure 26. Finite Element Type

$$\left\{ \begin{array}{l} x_9 = \frac{1}{8}\sum_{i=1}^{8} x_i \\ y_9 = \frac{1}{8}\sum_{i=1}^{8} y_i \\ z_9 = \frac{1}{8}\sum_{i=1}^{8} z_i \end{array} \right\} \qquad (4.11)$$

Unstructured grids are accomplished by nodes collapsing to form prisms, tetrahedrons, etc. as shown in Figure 27.

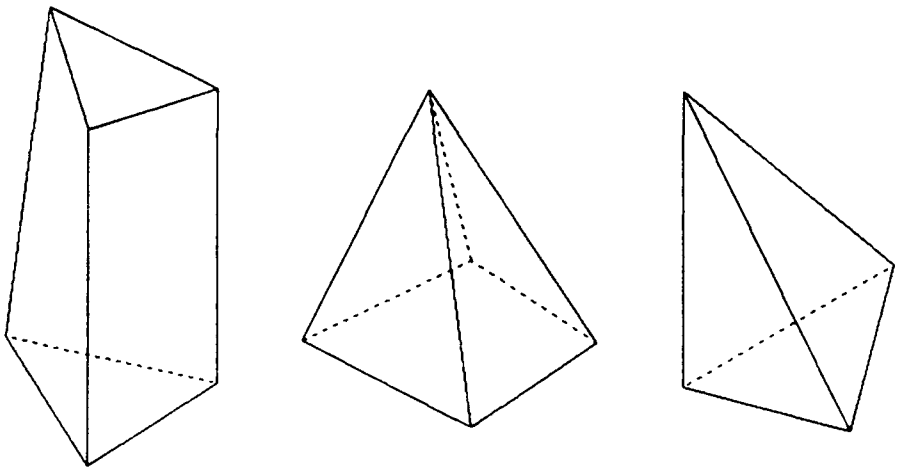


Figure 27. Other Shapes

Isoparametric Element

The general versions of Equations 4.4 and 4.5 will now be given. First, the general finite element of Figure 26 is mapped to a 2- by 2- by 2-unit cube in (ξ, η, ζ) computational type space, where

$$\left\{ \begin{array}{l} -1 \le \xi \le 1 \\ -1 \le \eta \le 1 \\ -1 \le \zeta \le 1 \end{array} \right\}$$

The values of ξ, η, and ζ at the corner nodes are either -1 or 1, and (0, 0, 0) at node 9. The ith interpolation function N_i for values of i from 1 to 8 becomes

$$N_i = \frac{1}{8}(1 + \xi_i\xi)(1 + \eta_i\eta)(1 + \zeta_i\zeta) \qquad (4.12)$$

where (ξ_i, η_i, ζ_i) are the (ξ, η, ζ) coordinates at point i. The interpolation function at point 9 is

$$N_9 = (1 - \xi^2)(1 - \eta^2)(1 - \zeta^2) \qquad (4.13)$$

Now h is interpolated by

$$h = \sum_{i=1}^{8} N_i h_i + N_9 h_9' \qquad (4.14)$$

where

$$h_9' = h_9 - \frac{1}{8}\sum_{j=1}^{8} h_i \qquad (4.15)$$

so that the matrix version again becomes as in Equation 4.5,

$$h = \{N\}^T \{h\}_e$$

Because $\{r\}_9$, where

$$\{r\} = \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}$$

is defined at the average point of the element, $\{r\}'$ at node 9 is zero. So,

$$\{r\} = \sum_{i=1}^{8} N_i \{r\}_i$$

A matrix form for this equation is

$$\{r\} = \begin{pmatrix} \{N\}^T & \{0\}^T & \{0\}^T \\ \{0\}^T & \{N\}^T & \{0\}^T \\ \{0\}^T & \{0\}^T & \{N\}^T \end{pmatrix} \begin{pmatrix} \{x\}_e \\ \{y\}_e \\ \{z\}_e \end{pmatrix}$$

In these equations,

$$\{h\}_e = \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9' \end{pmatrix} \quad \{x\}_e = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ 0 \end{pmatrix} \quad \{y\}_e = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ 0 \end{pmatrix} \quad \{z\}_e = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \\ z_8 \\ 0 \end{pmatrix}$$

Also, {0} is a zero vector. Let the above equation for {r} be written as

$$\{r\} = [N]\,[r]_e$$

where

$$[N] = \begin{pmatrix} \{N\}^T & \{0\}^T & \{0\}^T \\ \{0\}^T & \{N\}^T & \{0\}^T \\ \{0\}^T & \{0\}^T & \{N\}^T \end{pmatrix} \qquad [r]_e = \begin{pmatrix} \{x\}_e \\ \{y\}_e \\ \{z\}_e \end{pmatrix}$$

Stiffness Matrix Formulation

The element stiffness matrix as given in Equation 4.10 is now derived. First, the general equation for the gradient is

$$\{i\} = \nabla h = \nabla\{N\}^T\{h\}_e$$

So from Equation 4.6,

$$[B] = \nabla\{N\}^T \tag{4.16}$$

Now we will use the operators

$$\nabla = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix} \qquad \nabla' = \begin{pmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \\ \frac{\partial}{\partial \zeta} \end{pmatrix}$$

to determine [B]. First, the Jacobian Matrix [J] is defined by

$$[J] = \frac{\partial(x, y, z)}{\partial(\xi, \eta, \zeta)}$$

This can also be written

$$\begin{aligned} [J] &= \nabla'\{r\}^T \\ &= \nabla'([N][r]_e)^T \end{aligned}$$

Expanding,

$$\begin{aligned} [J] &= \nabla'(\{N\}^T\{x\}_e \quad \{N\}^T\{y\}_e \quad \{N\}^T\{z\}_e) \\ &= \nabla'\{N\}^T(\{x\}_e \quad \{y\}_e \quad \{z\}_e) \end{aligned}$$

The matrix,

$$[P] = \nabla'\{N\}^T \tag{4.17}$$

has only ξ, η, and ζ variables, so it can be easily evaluated. Therefore,

$$[J] = [P]\,(\{x\}_e \;\; \{y\}_e \;\; \{z\}_e) \tag{4.18}$$

Next, using the chain rule of differentiation,

$$\nabla' = [J]\,\nabla$$

For elements that are not skewed or ill-conditioned, [J] is well defined. Therefore,

$$\nabla = [J]^{-1}\nabla'$$

[B] then becomes

$$[B] = [J]^{-1}\nabla'\{N\}^T$$

Using Equation 4.17 we get

$$[B] = [J]^{-1}[P] \tag{4.19}$$

[B] is now completely defined in terms of constants and computational coordinates.

The element stiffness matrix as given in Equation 4.10 can now be evaluated by integrating in computational space as follows:

$$[K]_e = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} [B]^T[k]\,[B]\,|J|\;d\xi\,d\eta\,d\zeta \tag{4.20}$$

Numerical Integration

Equation 4.20 must be integrated numerically by a Gauss Quadrature formula as shown:

$$I = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} f(\xi, \eta, \zeta)\, d\xi d\eta d\zeta$$

$$\approx \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} w_i w_j w_k f(\xi_i, \eta_j, \zeta_k)$$

n is a specified integer, and from this choice the w's and (ξ_i, η_j, ζ_k)'s are set. n = 2 integrates a cubic equation exactly, and this typically gives enough accuracy. However, n = 4 is used in the stiffness matrix computations because of the special needs of the unconfined flow algorithm discussed later.

Decomposition

The 9 by 9 stiffness matrix is next reduced to an 8 by 8 matrix by eliminating terms related to the ninth or internal node. This is done by partitioning the 9 by 9 equation,

$$[K]_9\{h\}_9 = \{Q\}_9$$

as follows:

$$\begin{pmatrix} [K]_8 & \{K\}_8 \\ \{K\}_8^T & K_{99} \end{pmatrix}\begin{pmatrix} \{h\}_8 \\ h_9 \end{pmatrix} = \begin{pmatrix} \{Q\}_8 \\ 0 \end{pmatrix}$$

Here $Q_9 = 0$ because no flow is considered to enter at the internal node. This matrix equation represents two equations where the second one can be solved for h_9 and substituted into the first one. This gives

$$h_9 = -\frac{1}{K_{99}} \{K\}_8^T \{h\}_8$$

and

$$([K]_8 - \frac{1}{K_{99}} \{K\}_8 \{K\}_8^T) \{h\}_8 = \{Q\}_8$$

So the actual stiffness matrix $[K]_a$ used for further computations is

$$[K]_a = [K]_8 - \frac{1}{K_{99}} \{K\}_8 \{K\}_8^T \qquad (4.21)$$

Discharge Velocity

The discharge velocity {v} for each element can be computed by the general version of Equation 4.8 as follows:

$$\{v\} = -[k] [B] \{h\}_e \qquad (4.22)$$

For the element type used in this application, discharge velocity is typically computed at the $(\xi, \eta, \zeta) = (0, 0, 0)$ point which is close to the centroid of the element. The element discharge velocity $\{v\}_0$ at this point thus becomes

$$\{v\}_0 = -[k] [B]_0 \{h\}_e \qquad (4.23)$$

where $[B]_0$ is the [B] matrix evaluated at $(\xi, \eta, \zeta) = (0, 0, 0)$.

Assemblage with Boundary Conditions

With the element stiffness matrices computed, one can now assemble a global set of equations

$$[K]_G\{h\}_G = \{Q\}_G \tag{4.24}$$

$\{h\}_G$ is the set of global heads at the active nodes that must be computed. If the boundary condition of specified head is applied to a node, it is not active because head is already known. $[K]_G$ is computed by assembling the element stiffness matrices taking into account the active nodes as follows:

$$[K]_G = \sum_{i=1}^{N_E} [K]_{ai}$$

where N_E is the number of elements and $[K]_{ai}$ is the ith element stiffness matrix as given in Equation 4.21.

$\{Q\}_G$ is the combination of specified flows at nodes plus those computed from a specified normal discharge velocity v_N. Let $\{Q\}_4$ be flows computed at four nodes of a face of an element in (ξ, η) computational space where v_N is applied. Also, let the shape functions $\{N\}_4$ now be given by

$$N_i = \frac{1}{4}(1 + \xi_i\xi)(1 + \eta_i\eta) \quad i = 1, 2, 3, 4$$

Then

$$\{Q\}_4 = \int_{-1}^{1}\int_{-1}^{1} v_N \sqrt{EG - F^2}\ \{N\}_4\, d\xi\, d\eta \tag{4.25}$$

where

$$E = \frac{\partial \{r\}^T}{\partial \xi} \frac{\partial \{r\}}{\partial \xi}$$

$$F = \frac{\partial \{r\}^T}{\partial \xi} \frac{\partial \{r\}}{\partial \eta}$$

$$G = \frac{\partial \{r\}^T}{\partial \eta} \frac{\partial \{r\}}{\partial \eta}$$

Now $\{r\}$ depends only on four nodes on the surface. Let these (x, y, z) coordinates be designated by $\{x\}_4$, $\{y\}_4$, and $\{z\}_4$, respectively. Then $\{r\}$ is simply

$$\{r\} = [N]_4 [r]_4$$

where

$$[N]_4 = \begin{pmatrix} \{N\}_4^T & \{0\}_4^T & \{0\}_4^T \\ \{0\}_4^T & \{N\}_4^T & \{0\}_4^T \\ \{0\}_4^T & \{0\}_4^T & \{N\}_4^T \end{pmatrix} \qquad [r]_4 = \begin{pmatrix} \{x\}_4 \\ \{y\}_4 \\ \{z\}_4 \end{pmatrix}$$

Also, $\{0\}_4$ is a zero vector of length 4. The expressions for E, F, and G are now easily evaluated. For instance

$$E = [r]_4^T \frac{\partial [N]_4^T}{\partial \xi} \frac{\partial [N]_4}{\partial \xi} [r]_4$$

All the terms in Equation 4.25 can now be evaluated, so the expression can be numerically integrated as before. Once the $\{Q\}_4$ are computed, they are assembled into the global Q vector $\{Q\}_G$.

All is now ready to solve Equation 4.24 for the unknown heads $\{h\}_G$. This is a symmetric banded system of simultaneous, linear equations that can now be solved. Once

the heads are known for each node the discharge velocities can be computed for each element using Equation 4.22. If the problem is a confined flow problem, the solution is now complete. However, for an unconfined flow problem this represents only the initial iteration. Details regarding unconfined flow problems are given next.

Unconfined Flow Problem

The unconfined flow problem requires an iterative process to determine a solution because of the free surface being unknown. We will now examine this problem in detail.

Description of the Problem

To see the complexity in two dimensions, consider the earth dam shown in Figure 28. Line segment DG is the free surface that is unknown. The exit point is point G, and segment GH is the surface of seepage. Notice that point G does not necessarily coincide with point H where the tail water starts. Line segment BCD is an equipotential line,

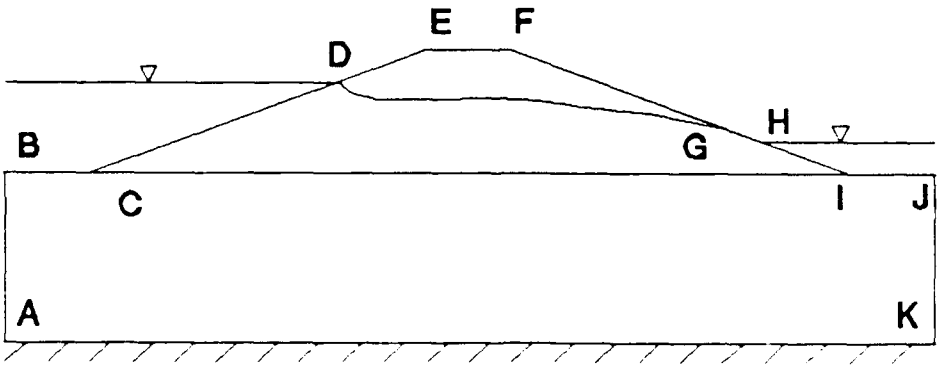


Figure 28. Unconfined Flow Problem

and since DG is a flow line at steady-state conditions, DG is perpendicular to BCD at D. HIJ is an equipotential line, and DEFG is removed from the problem. Points A and B are moved far enough left to where the effects of the dam are not felt, and the same is done to points J and K, except they are moved far to the right.

The exit point becomes an exit line in three dimensions. Further, in an aquifer with wells, there are any number of independent tailwater levels, exit lines, and surfaces of seepage. Nevertheless, it is the goal of this dissertation to allow the user to solve all these problems without restriction of grid. It is also extremely tedious to have to give an initial guess of the free surface, especially when there could be any number of them in 3-D, so no initial guess is to be required.

Geometric Considerations

Figure 29 shows a portion of an FEM grid with the free surface ABCDE cutting through it. Point A is at the headwater level, and point F is at the tailwater level.

Reshaped Elements

First, the free surface cuts the grid so that some elements are completely out of the problem after a few iterations, and others have new shapes. Some become very small or skewed, while others have five sides. In three dimensions these different cases become even more complicated.

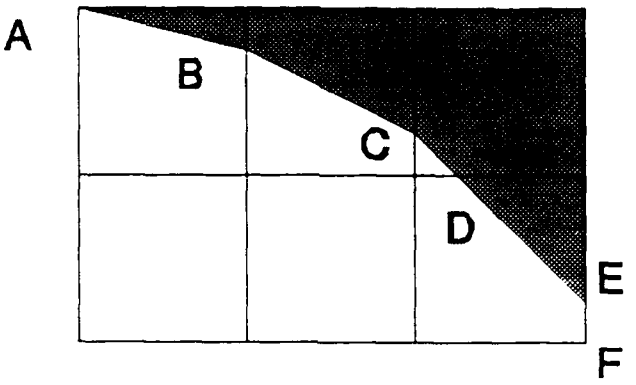


Figure 29. FEM Grid with Free Surface

Clearly, an adaptive grid scheme or another procedure is more advantageous.

Exit Point

The exit point E also causes a problem. Points B, C, and D can be considered to have no flow entering at these points since the free surface is a flow line. Previous work has also given this no-flow condition to the exit point E. This, however, is incorrect, and if done this way, leads to a wrong exit point. To understand, consider Figure 30 where the arrows show the direction of flow. Flow is lumped at a node by attributing to that node the flow in the vicinity of that node. Although no flow is added to point E from the top, there is definitely some flow from the surface of seepage part below the node. This situation becomes even more complex in three dimensions. Clearly, it would be better to

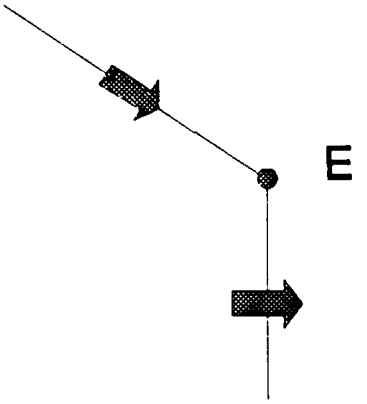


Figure 30. Exit Point E

find an algorithm that completely avoids these problems, at least during the iteration process.

Computational Procedure

After trying several algorithms, the one best fulfilling the goals of this dissertation is now outlined:

1. Compute and store $[K]_{a0}$ for each element.
2. Assemble $[K]_u$ and $[K]_{G0}$.
3. Solve $[K]_{G0}\{h\}_{G0} = \{Q\}_{G0}$ for $\{h\}_{G0}$.
4. Compute $\{Q\}_u = [K]_u\{h\}_{G0}$.

For j = 1, 2, 3, ...

5. Check for convergence.
6. Compute $[K]_{aj}$ for each element.
7. Compute $\{\Delta Q\}_{ej} = [K]_{aj}\{h\}_{e,j-1}$ and assemble
8. Compute different global stiffness matrix
9. Solve $[K]_{G'j}\{\Delta h\}_{Gj} = -\{\Delta Q\}_{Gj}$ for $\{\Delta h\}_{Gj}$.
10. Update $\{h\}_{Gj} = \{h\}_{G,j-1} + \{\Delta h\}_{Gj}$.

End j

11. Compute the final position of the free surface.

12. Compute the final element discharge velocities.

Each part will now be described in detail.

Compute and Store $[K]_{a0}$ for Each Element

The first thing that is done is to compute the stiffness matrices for each element. Although possibly modified in subsequent iterations, the initial material properties (permeabilities) are used here. The element stiffness matrices are stored and used later in the iteration process.

Assemble $[K]_u$ and $[K]_{G0}$

The unmodified stiffness matrix $[K]_u$ is assembled by using both active and inactive nodes in the assemblage process. Another way of saying this is that the boundary conditions are not considered. $[K]_{G0}$ is determined by either assembling the global stiffness matrix using only active nodes (those where heads are not specified) or modifying $[K]_u$ for boundary conditions.

Solve $[K]_{G0}\{h\}_{G0} = \{Q\}_{G0}$ for $\{h\}_{G0}$

This is a standard system of banded, simultaneous, linear equations that are solved for the heads at each node.

Compute $\{Q\}_u = [K]_u\{h\}_{G0}$

The flows at internal nodes are typically zero. One exception is when a source or sink is specified. The nodes where flow is specified will give the same answer. The nodes where head is specified will have their unknown flows computed by this operation.

Check for Convergence

The basic idea is to iterate until the free surface has stopped moving. Various criteria can be used to accomplish this with the one that proved best being now presented. First, compute a small number ϵ,

$$\epsilon = 0.001(z_{max} - z_{min})$$

where z_{max} is the maximum z, and z_{min} is the minimum z. Next, compute the maximum change in head for this iteration for all nodes that are still in the flow region as follows:

$$\Delta h_{max} = \underset{i}{MAX} \left| (\{\Delta h\}_{G,j-1})_i \right|$$

where i is the ith eligible node. When

$$\Delta h_{max} \leq \epsilon$$

for two straight iterations, the solution has converged. The reason two iterations are used is that for some rare problems the criteria must be applied twice in a row to achieve true convergence. Otherwise, false convergence occurs after two or three iterations.

Compute $[K]_{aj}$ for Each Element

As shown in Figure 29, the free surface is above some elements, goes through others, and is completely below yet others. For elements that are completely inside the flow region, $[K]_{aj}$ is read from the disc and used as is. For those completely above the region, the strategy of using a reduced element stiffness matrix is used. This can be either in steps or all at once. The reduction factor of 0.001 is used here. This is equivalent approximately to saying that no flow is in the element. Figure 31 shows a 2-D example of what is done for elements where the free

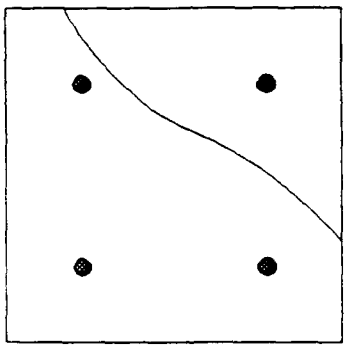


Figure 31. Free Surface through 2-D Element

surface crosses the element when two interpolation points in each direction are used. Basically, the numerical integration process in this case involves integrating over a discontinuity. The pressure head is computed at each integration point used in integrating Equation 4.20, and, if less than zero, the reduced value of permeability tensor is used. Also, a higher number of integration points are used since a

discontinuity is involved. So the 3-D implementation has the total head computed at each of 4 by 4 by 4 numerical integration points as

$$h(\xi_i, \eta_j, \zeta_k) = \{N(\xi_i, \eta_j, \zeta_k)\}^T\{h\}_e$$

where i, j, and k range from 1 to 4. The pressure head h_p for an arbitrarily specified datum h_D is

$$h_p = h + h_D - z$$

Compute $\{\Delta Q\}_{ej}$ and $\{\Delta Q\}_{Gj}$

For an internal node the flow should be zero. If it is not, this indicates that flux is going across the boundary at that node, and it is part of a moving free surface. The technique used is to first compute element flows $\{\Delta Q\}_{ej}$ by

$$\{\Delta Q\}_{ej} = [K]_{aj}\{h\}_{e,j-1}$$

and then assemble them into a change in flow for each node designated by $\{\Delta Q\}_{Gj}$. For an external node where the flow Q_E is specified, the change in flow $\Delta Q'$ becomes

$$\Delta Q' = \Delta Q - Q_E$$

Here, ΔQ is the value from the array $\{\Delta Q\}_{Gj}$ for that node.

Compute Different Global Stiffness Matrix $[K]_{G'j}$

The changes in head $\{\Delta h\}_{Gj}$ could be computed using the equation

$$[K]_{Gj}\{\Delta h\}_{Gj} = -\{\Delta Q\}_{Gj} \qquad (4.26)$$

where $[K]_{Gj}$ is the global stiffness matrix assembled from the $[K]_{ej}$. However, since a steady-state solution is desired, it is expedient to employ the aerospace engineers' strategy that states for a numerical expression solving for Δh

$$(NUMERICS)\,(\Delta h) = PHYSICS$$

Now the physics require that the right-hand side of Equation 4.26 go to zero. But the numerics of the left-hand side can be simplified to achieve this goal by using the unmodified stiffness matrix $[K]_u$ (which has already been computed) and applying the current boundary conditions to obtain a different global stiffness matrix $[K]_{G'j}$. As will now be explained using Figure 32, the only place where

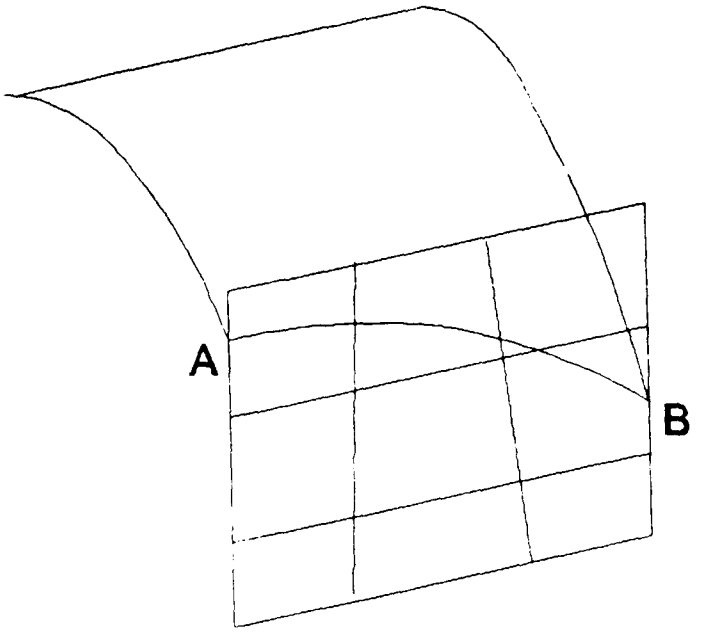


Figure 32. Surface of Seepage

boundary conditions change is on the surface of seepage. Line segment AB is an exit line on an exit face, which is shown only with the external sides of the brick elements at the exit face. Each node below the line of seepage and above or on the tailwater area will have a negative flow, since flow is leaving the system at that point. So when such a node is encountered, its pressure head is zero, and its boundary condition for head is

$$h = z - h_D$$

Now, if in the course of iteration the flow for this node becomes positive, then the free surface has fallen below this node, and its boundary condition must be changed. What is done is illustrated in the 2-D sketch shown in Figure 33 where the region above the exit point is made impervious. So the node with the status change will have its boundary condition shifted from specified head to the no-flow boundary condition.

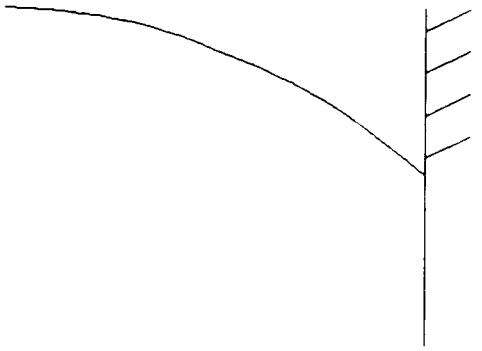


Figure 33. 2-D Exit Point Analysis

A node can also be in the no-flow, impervious condition and need to change back to a node below the exit line. This is, in fact, a case of a rising exit line. What is done is to check the currently impervious node on the surface of seepage for the condition

$$h \geq z - h_D$$

If this condition is passed, then the node is changed back to a specified head node with the boundary condition

$$h = z - h_D$$

again applied.

Compute $\{\Delta h\}_{Gj}$

Let the unmodified stiffness matrix $[K]_u$ which has now been modified at iteration j by the current boundary conditions be designated as $[K]_{G'j}$. Then the actual system of equations used to compute the change in head at iteration j is

$$[K]_{G'j}\{\Delta h\}_{Gj} = -\{\Delta Q\}_{Gj} \qquad (4.27)$$

Equation 4.27 is a symmetric, positive definite, banded system of simultaneous linear equations that can now be solved for $\{\Delta h\}_{Gj}$.

Update $\{h\}_{Gj}$

The active nodes can now have their total heads updated as follows:

$$\{h\}_{Gj} = \{h\}_{G,j-1} + \{\Delta h\}_{Gj} \qquad (4.28)$$

One iteration is now complete, and the cycle is repeated. The process is continued until convergence is achieved.

Determine Final Position of Free Surface

The determination of the final position of the free surface for a structured grid algorithm is rather straightforward. However, the thrust of this work is to do a free surface problem without any required structure at all to the grid. The determination of the free surface in this case is significantly more difficult, especially at the exit line. First, consult Figure 34 to understand the complexities away from the exit line. Here two tetrahedral elements, ABEC and ACED, of an unstructured 3-D grid are shown with the free surface denoted by the cross-hatched plane crossing these elements. First, the reason the free surface crosses these elements is that the pressure head at point A is less than zero, and at points B, C, D, and E the pressure head is greater than zero. Now it is best to keep the same data structure (FEM mesh) for the purpose of computing discharge velocities at the elements and providing output to post-processor programs. So one strategy is to "clip" an affected element by the free surface to create two elements. Then one element would be completely above the free surface, and one would be completely below the free surface. However, this leads to significant complications. Notice,

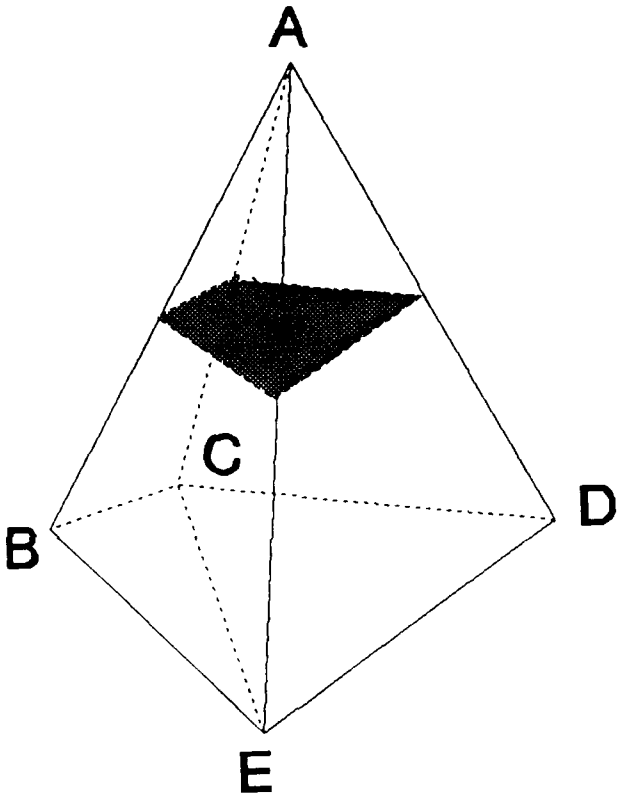


Figure 34. Free Surface across Grid

first of all, that no problem exists in Figure 34 where each four-sided piece is changed into a four-sided piece and a five-sided piece. However, consider the effect of a six-sided element being clipped by the free surface with the piece left below the free surface shown in Figure 35. What is left is a seven-sided piece, no longer conforming to the allowable shapes of the eight-node brick element. This problem could be alleviated by dividing the seven-sided figure into acceptable pieces. However, the variety of

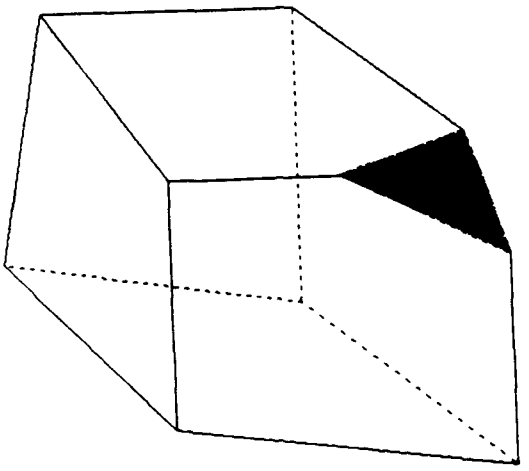


Figure 35. Seven-Sided Piece

unacceptable shapes warranted the search for another solution. Point A in Figure 34 with a negative pressure head was moved to a point on the free surface where the pressure head is zero. This, however, creates a new dilemma. Any point in the cross-hatched surface representing the free surface has zero pressure head, so some criteria must be devised to decide which point will be chosen.

After trying different algorithms, the one that works well and was implemented will now be given:

I. Identify a node (such as point A in Figure 36) with pressure head less then zero.

   A. Find a line segment emanating from A so that the second point of the line segment (such as point B in Figure 36) has pressure head greater than zero.

      1. Compute the value of the parameter s

$$0 \leq s \leq 1$$

with s = 0 at point A and s = 1 at point B where the pressure head is zero (point O in Figure 36). Linear interpolation gives

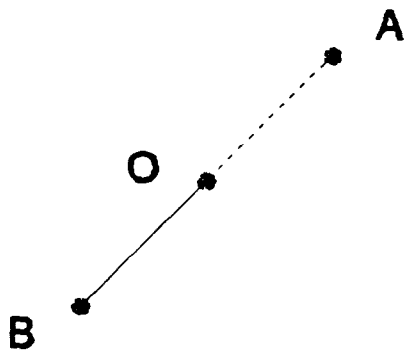


Figure 36. Line Segment AB

$$s_O = \frac{h_{pA}}{h_{pA} - h_{pB}} \qquad (4.28)$$

where h_{pA} is the pressure head at point A, h_{pB} is the pressure head at point B, and s_0 is the value of s at point O.

2. Compute the new position of point A along AB so it resides on the free surface by

$$\{r\}_O = (1 - s_O)\{r\}_A + s_O\{r\}_B$$

where $\{r\}_Q$ is the new position of point A, $\{r\}_A$ is the original position of point A, and $\{r\}_B$ is the position of point B.

3. Compute the distance squared from point A to point O. Save that number with the current position of the free surface.

B. Repeat step A for other line segments starting with point A. Keep the one $\{r\}_0$ with the smallest distance squared value.

II. Process all other node points just above the free surface that need to be moved in the same way as in step I.

A node point on the surface of seepage cannot be done exactly this way. The reason is that the value of s_0 cannot be computed. To understand, first consider the 2-D case illustrated in Figure 37. Line segment AB is a line on the surface of seepage, and E is the exit point to be computed. Now AB contains an exit point because (1) the nodes have surface of seepage boundary conditions and (2) node A was flagged as an impervious node and node B was flagged as a specified head node at the last iteration cycle before convergence. Points a, b, and c are the three most adjacent free surface nodes that have been moved from their original position to their new positions, respectively, to be used for computing point E.

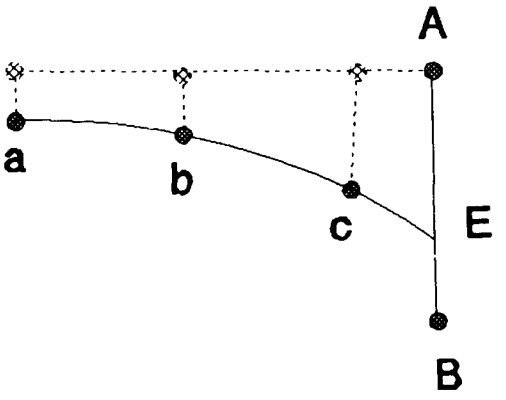


Figure 37. Exit Point Computation

Exit point E must be determined by extrapolation. What is used is a difference formulation using points 1 through

3. First, the slope at point a is computed by the backward difference expression

$$m_a = \frac{(y_b - y_a)}{(x_b - x_a)}$$

In a similar manner the slope at point b is

$$m_b = \frac{(y_c - y_b)}{(x_c - x_b)}$$

From these results the change in slope c_m can be computed by

$$c_m = \frac{(m_b - m_a)}{(x_b - x_a)}$$

We can now compute the slope at point c using

$$m_c = c_m(x_c - x_b) + m_b$$

With x_c, y_c, and m_c all known this line can be intersected with line segment AB to obtain the intersection point E. This is accomplished by substituting into the equation

$$y = y_c + m_c(x - x_c)$$

the parametric equation

$$\{r\} = (1 - s)\{r\}_A + s\{r\}_B$$

and solving for s. If s does not lie between zero and one, the intersection point occurs outside the range of line segment AB, and point B is taken as the exit point.

The 3-D case is much more difficult. Figure 38 shows a plan view of a portion of the free surface just computed

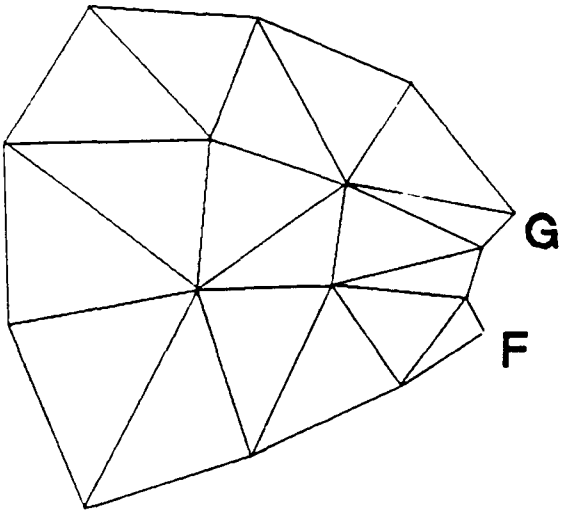


Figure 38. 3-D Free Surface

with the exit line FG needing to be computed. What is significantly more difficult is the need to extrapolate off an unstructured surface to get the intersection of that surface with the surface of seepage yielding the points on the exit line FG. After some analysis of these difficulties, it was decided to take another approach. Figure 39 shows an almost vertical exit face where it has been determined that point A must move down on this surface to become part of the exit line. Now at the last iteration, point B has the surface of seepage boundary condition

$$h = z - h_D$$

and point A is denoted as impervious with a computed head value

$$h_A < z - h_D$$

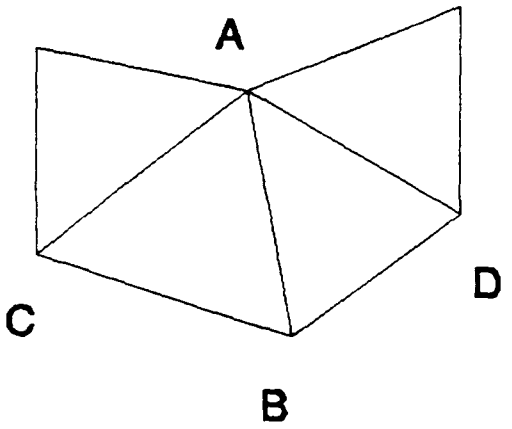


Figure 39. Exit Face

The amount point A must move down line segment AB is determined by computing a value for the parameter s. First, compute s_0 by

$$s_0 = \frac{h_A + h_D - z_A}{z_B - z_A} \tag{4.29}$$

The final value of s is computed from

$$s_1 = (1 - \gamma)\, s_0 + \gamma \tag{4.30}$$

with

$$0.5 \leq \gamma \leq 1$$

From experience the value used in this work is 0.8. The new position of point A on AB is computed as before

$$\{r\}_1 = (1 - s_1)\{r\}_A + s_1\{r\}_B$$

As in the other free surface computations, point A can move in more ways than along line segment AB. In Figure 39,

for instance, line segments AC and AD must also be considered. As before, the line segment is kept with the smallest distance of movement of point A from its original position.

Compute Final Element Discharge Velocities

Now that the nodes near the free surface have been moved to be on the free surface, the element data must be modified so that all elements are either completely above the free surface or completely below it. Figure 40 shows an originally square mesh with the new shapes of the elements conforming to the free surface shown. Those elements above the free surface will be given a value for discharge velocity of zero, and those below the free surface will have their discharge velocities computed using Equation 4.23. Additional care must also be taken to ensure that the newly formed elements that become skewed are handled properly. They can be either removed from the grid or kept and given zero discharge velocity. In this work they are kept and given zero values of discharge velocity for compatibility with postprocessing programs.

Achieving the newly formed grid as shown in Figure 40 is more difficult than it first appears. First, consider the 2-D case where Figure 41 shows the three steps used in the process. Step 1 consists of determining the position of the free surface nodes on the proper line segments (the dots in Figure 41). In step 2 the free surface nodes are moved

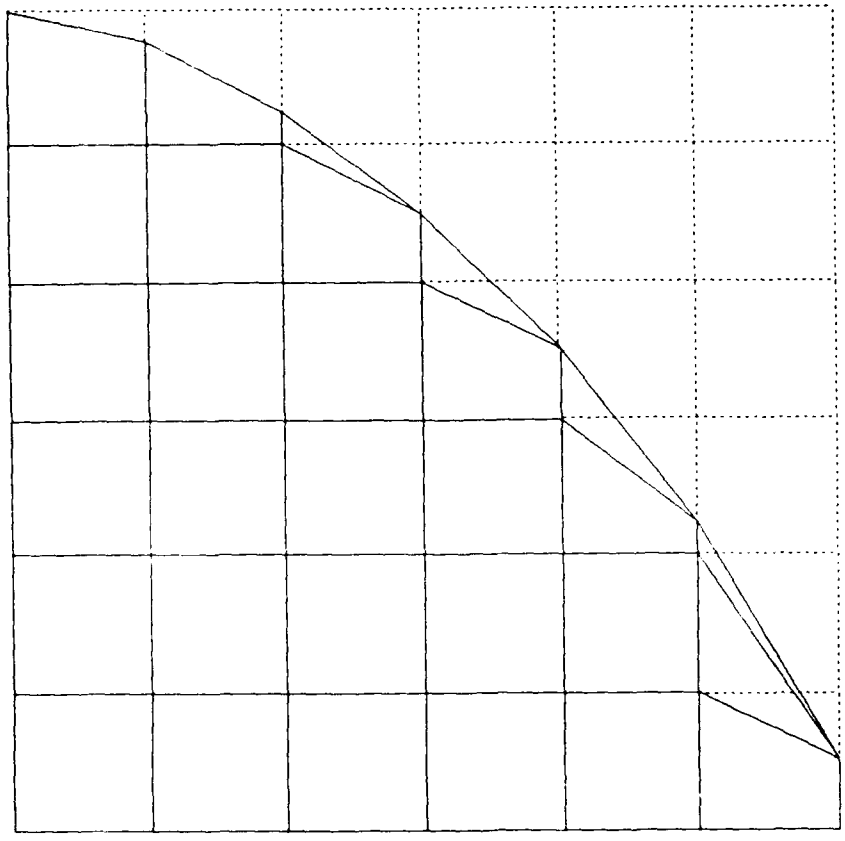


Figure 40. New Element Shapes

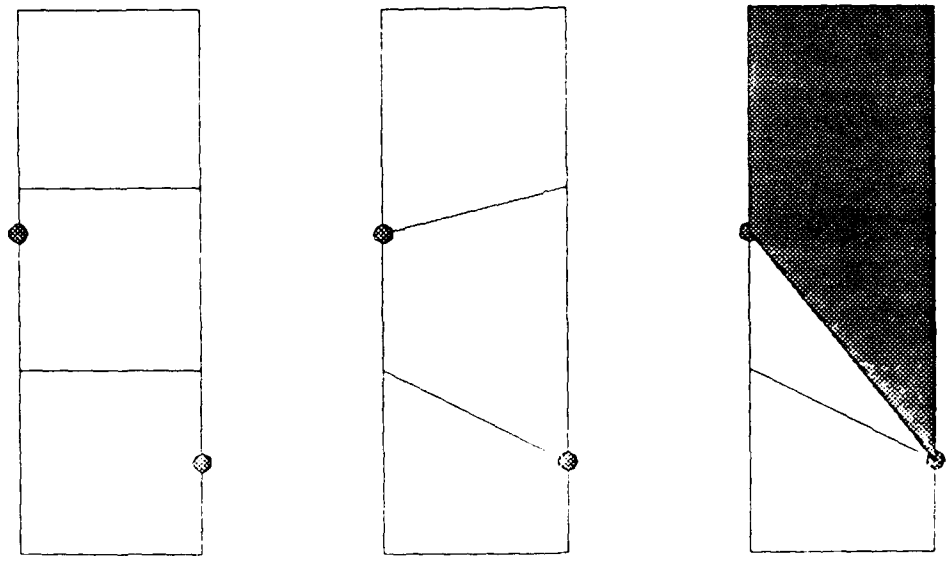


Figure 41. Three-Step Process

to their new positions. Step 3 is the most complicated in that additional movement of nodes is required to get elements either completely below or completely above the free surface.

The 3-D version of this problem is even more complicated. This is because of the many different shapes the 3-D brick element can take as shown in Figure 27. Also, because of the general nature of the 3-D element, establishing what is on top and what is on bottom becomes blurred. However, as illustrated in Figure 42, a 3-D version of the algorithm presented above was developed as part of this work. Figure 42 shows a brick element with free surface nodes

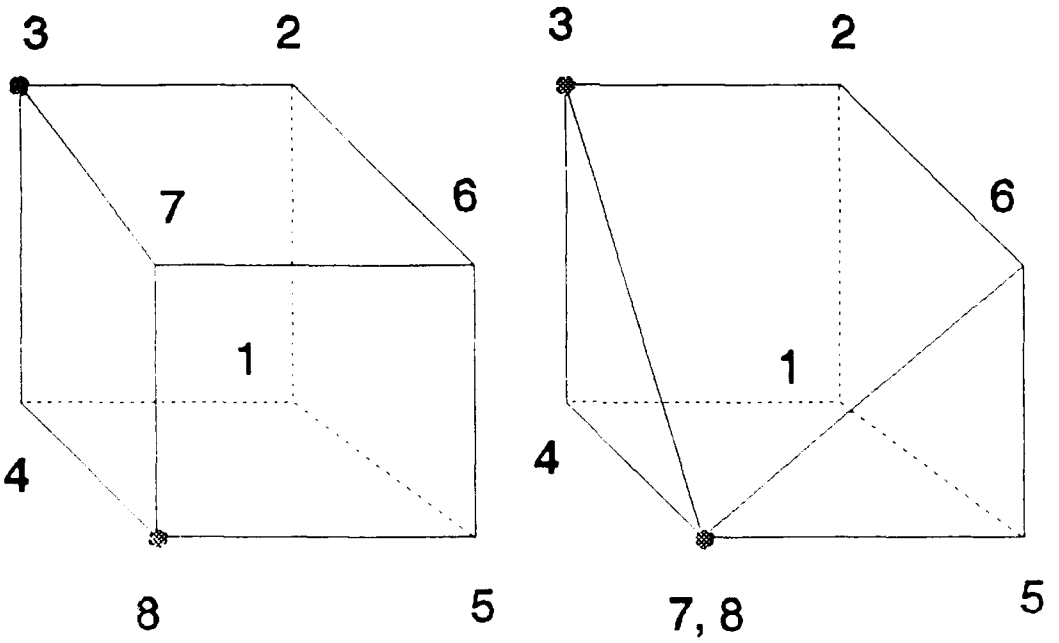


Figure 42. 3-D Version

occuring exactly at nodes 3 and 8. The first plot shows the element before any change, and the second plot shows the element after it was modified to fit the free surface. Step 3 of the algorithm collapsed node 7 down to node 8. In fact, the first step 3 algorithm considered checks each of the six faces of an element to see if any two nodes on a diagonal of a face are bath free surface nodes. If two exist, then any other node on the face having a head value

$$h < z - h_D$$

will be collapsed to one of the diagonal nodes.

This algorithm, however, does not always work. The 2-D version of this algorithm, for instance, works on all vertical columns of six elements each in Figure 40 except the last one. The isolated part of the grid where the problem exists is shown in Figure 43. Figure 44 shows the three step sequence and how it breaks down. The middle element marked by the X does not have two free surface nodes on a diagonal, so the third step is not completed. Therefore, a higher order iterative algorithm must be implemented.

The following algorithm was developed as part of this work:

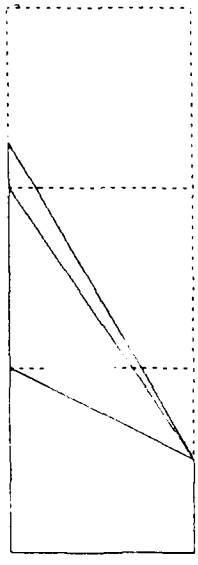


Figure 43. Isolated Piece

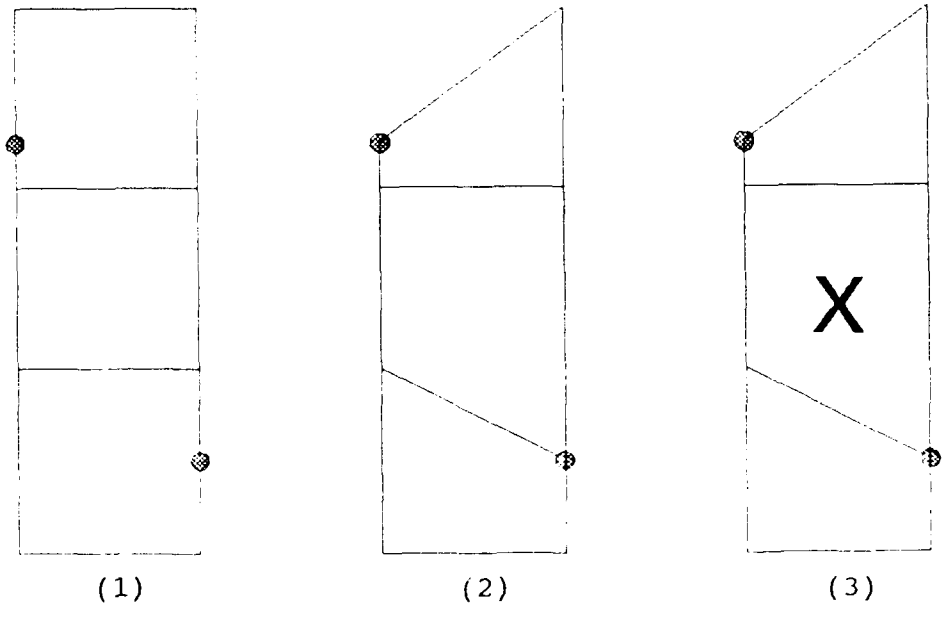


Figure 44. Three Steps for Isolated Piece

```
Repeat until no change occurs

   For n = 1 to the number of elements

      If the free surface intersects the element, then

         For i = 1 to 12  /* Test all line segments. */

          If a free surface node is connected to a node
             above the free surface, then

               Change the coordinates and head of the
               node above the free surface to that of the
               free surface node.

               Flag the moved node as a free surface
               node, but as an artificially generated
               one.

            End If

         End i

      End If

   End n

End Repeat
```

The algorithm can be utilized in two ways:

1. Adjust only those nodes immediately above the free surface and leave the other nodes alone.

2. Adjust those nodes immediately above the free surface and have the nodes farther above the free surface to collapse to the free surface.

Appendix A contains a FORTRAN listing and description of the free surface algorithms described in this chapter with implementation of Option 2 for utilizing the algorithm.

CHAPTER V

THREE-DIMENSIONAL SEEPAGE PACKAGE AND GROUNDWATER MODEL

Introduction

A three-dimensional seepage package and groundwater model was developed that concentrated on three areas:

1. Sophisticated numerics to allow both structured and unstructured grids with a minimum of input for free surface problems.
2. Interface to state-of-the-art grid generation technology.
3. Interface to state-of-the-art scientific visualization technology.

The numerics were discussed in detail in Chapter IV. This chapter describes the components of the 3-D model.

Grid Generation

EAGLE, written originally for aerospace engineering applications (computational fluid dynamics (CFD)) to generate structured grids for finite volume solvers, was used as the primary grid generation tool in this work. It has extensive algebraic grid generation capabilities, as well as state-of-the-art elliptic smoothing capabilities. However, since the FEM was the computational tool used, partly unstructured of totally unstructured grids may also be used with the developed computer program.

Conversion to FEM Format

The finite volume grid from EAGLE with potentially several blocks must be converted to the final data needed by the FE program. Five things are involved:

1. Getting title, datum, and material property information.
2. Applying boundary conditions.
3. Combining the different blocks into one FEM grid.
4. Applying bandwidth minimization.
5. Writing an output file containing this information.

Appendix B contains a description of the program written to accomplish these tasks.

Getting Data

Getting title, datum, and material property information is a simple matter of prompting the user for this information.

Applying Boundary Conditions

The EAGLE finite volume grid being structured makes applying boundary conditions much easier than usual. What is done is to ask for the following data:

IBLOCK, I1, J1, K1, I2, J2, K2, IBC, BV

where the meanings of the variables are as follows:

IBLOCK - Block number.

I1 - First i value.

J1 - First j value.

K1 - First k value.

I2 - Second i value.

J2 - Second j value.

K2 - Second k value.

IBC - Boundary value code.

BV - Boundary value.

The possible values of IBC and BV are given:

IBC Value	Boundary Value
1	Specified head.
2	Exit face. Head is computed by $h = z - h_d$.
12	Border of exit face and tail-water level. Head is computed the same as IBC = 2.
-1	Specified flow.
-2	Specified discharge velocity.

What this allows the user to do is to apply a boundary condition on an entire face. For example, consider the quadrilateral earth dam represented by one block shown in Figure 45. The i index of the block varies in the horizontal direction, the j index varies into the paper, and the k index varies along lines such as ABE and IG. Suppose

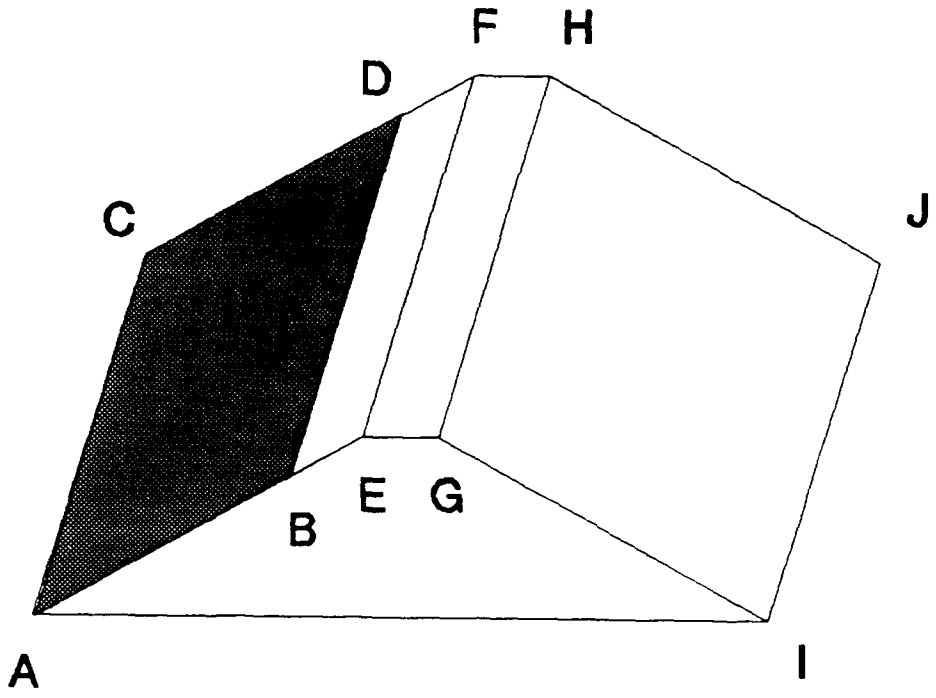


Figure 45. Quadrilateral Earth Dam

this is a 101 by 101 by 101 points block. Also, let the (x, y, z) coordinates and (i, j, k) values be as follows:

Letter	x	y	z	i	j	k
A	0	0	0	1	1	1
B	80	0	12	1	1	81
C	0	200	0	1	101	1
D	80	200	12	1	101	81
E	100	0	15	1	1	101
F	100	200	15	1	101	101
G	110	0	15	101	1	101
H	110	200	15	101	101	101
I	210	0	0	101	1	1
J	210	200	0	101	101	1

A headwater value of 12 ft can now be specified by the data:

IBLOCK	I1	J1	K1	I2	J2	K2	IBC	BV
1	1	1	1	1	101	81	1	12

In a similar manner, the boundary conditions of the exit face are defined by the two lines of data:

IBLOCK	I1	J1	K1	I2	J2	K2	IBC	BV
1	101	1	2	101	101	101	2	-
1	101	1	1	101	101	1	12	-

The - for Bv indicates that no data is needed, but a zero (0) must be supplied for the sake of completeness of data.

Combining Different Blocks

The process of combining different blocks into a single finite element mesh consists of the following:

1. Obtaining a single set of consecutively numbered nodes and elements.
2. Removing duplicate nodes and modifying the node numbering and element connectivity matrix to reflect the changes.

Note that duplicate nodes can occur in an O type grid along the cut as well.

Applying Bandwidth Minimization

There are different optimization schemes with varying degrees of sophistication. Some rearrange the node numbers while others modify the element numbering. It was decided to keep the element numbers the same in this program to allow the ability to output data to plotting programs like FAST. Therefore, a bandwidth minimization algorithm was

implemented to modify the node numbers. Because memory is plentiful on the Cray, a straightforward algorithm was used as follows:

1. Create an adjacency table stating what other nodes are connected to a given node and how many.

2. Place the new node 1 at various places on the grid. Then number the adjacent nodes to the new node 1 as 2, 3, 4, etc.

3. Go to the new node 2 and number the adjacent nodes to that node which have not been numbered as 5, 6, 7, etc. Continue this process until all the nodes have been numbered.

4. After trying as many of the possible positions for the new node 1 as desired, keep the one with the smallest bandwidth.

Writing an Output File

The last thing to be done is to write an output file containing the FEM grid. This includes all the preliminary data, boundary condition information, and discharge velocity data.

Cray Version of FEM Program

The finite element program was converted to the Cray YMP with as much vectorization as possible being accomplished. The solution of the banded system of equations was solved by using an out-of-core solver with blocks of size 1250. The forward loop of the Gauss Elimination algorithm was vectorized, but the back substitution loop was not. Nevertheless, a problem having 10,644 nodes that took approximately 12 hr on the Silicon Graphics Power Series

4D/220 GT took 3 min, 20 sec on the Cray YMP 6/128. Even more savings can be attained by using an in-core solver and further vectorization. The initial savings were so dramatic that this enhancement will be done at a future date.

Scientific Visualization

An output file containing the results was written to link with various scientific visualization tools. Of course, this file differs depending on which tool is used. Appendix C contains a subroutine that writes a file for Flow Analysis Solver Toolkit (FAST) (Bancroft, Kelaita, McCabe, Merritt, Plessel, Globus, and Semans 1991). The output is similar to that required by PLOT3D or Scientific Visualization Systems (program SSV) sold by Sterling Software. However, the data must be converted to a binary file written in C to be input to SSV or the beta version of FAST.

Conversion from Element Data to Node Data

The discharge velocities are computed for a given element at the average value of the (x, y, z) coordinates of the nodes of that element. However, to link with finite volume scientific visualization tools, values are needed at each node. Consider Figure 46 to visualize the process used to get node values. Discharge velocity at node n is computed by some type of average of the discharge velocities

from points A, B, and C. The program in Appendix B uses a straight average. However, another popular choice is a weighted "one over distance squared" average given by

$$\{v\}_n = \frac{1}{D}\left(\frac{\{v\}_A}{d_{An}^2} + \frac{\{v\}_B}{d_{Bn}^2} + \frac{\{v\}_C}{d_{Cn}^2}\right)$$

where d_{An} is the distance between point A and node n, d_{Bn} is the distance between point B and node n, and d_{Cn} is the distance between point C and node n. D is given by

$$D = \frac{1}{d_{An}^2} + \frac{1}{d_{Bn}^2} + \frac{1}{d_{Cn}^2}$$

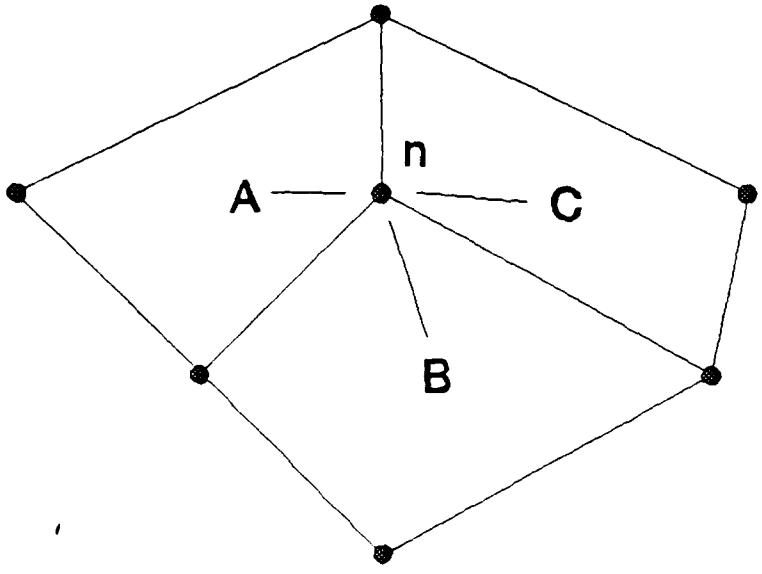


Figure 46. Element to Node Conversion

Conversion from Finite Element to Finite Volume Format

Now that values of head and velocity exist at each node, the block data structure must be reproduced to output a results file for FAST. The complication is that node numbers are scrambled from bandwidth minimization, nodes are eliminated when the blocks are combined, and the block sizes are discarded when creating the FEM grid. The last dilemma is solved by writing an output containing the block sizes. Further, since the element numbers have been preserved in the bandwidth minimization process and the elements were created in order of input, the element connectivity matrix can be used to write the output file. To understand, consider Figure 47 showing a 3- by 3- by 3-point block. First,

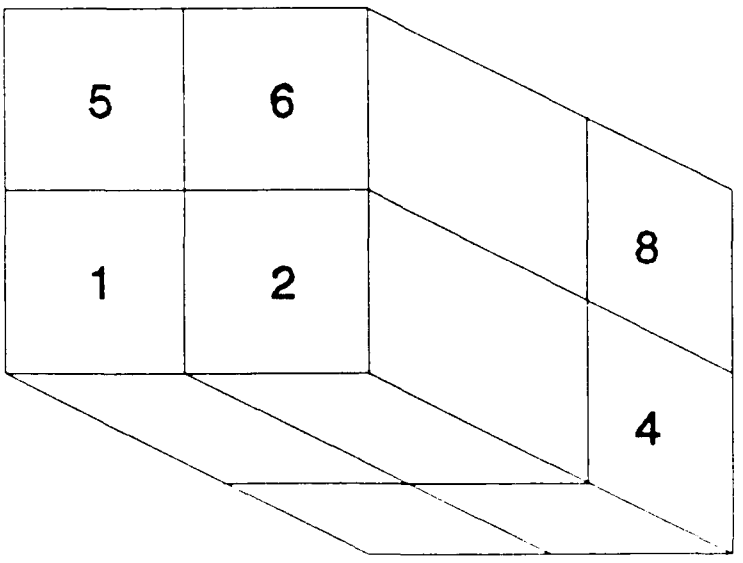


Figure 47. Block

there are 2 × 2 × 2 elements. So a triple loop with i2 = j2 = k2 = 3 and mstart = the element number just before the block begins can be set up as follows:

```
mm = mstart
For k = 1 to k2-1
   For j = 1 to j2-1
      For i = 1 to i2-1
         mm = mm + 1
         Output results for the 1st node of element m.
      End i

      /*  Do last one on i row.  */

      Output results for the 2nd node of element m.
   End j

   /*  Do last row in i, j plane.  */

   mm = mm - i2 + 1
   For i = 1 to i2-1
      mm = mm + 1
      Output results for the 4th node of element m.
   End i
   Output results for the 3rd node of element m.
End k

/*  Do last i, j plane.  */

mm = mm - (i2-1) * (j2-1)
For j = 1 to j2-1
   For i = 1 to i2-1
      mm = mm + 1
      Output results for the 5th node of element m.
   End i
   Output results for the 6th node of element m.
Erd j

/*  Do last row of last i, j plane.  */

mm = mm - i2 + 1
For i = 1 to i2-1
   mm = mm + 1
   Output results for the 8th node of element m.
End i
Output results for the 7th node of element m.
```

This process can be combined into a single set of loops by use of a variable (IPICK in the listing below) that selects which node of the given element should be processed. A FORTRAN implementation of the procedure follows:

```
      DIMENSION IPICK(2, 2, 2)
      DATA IPICK /1, 2, 4, 3, 5, 6, 8, 7/
C
      IROW = I2 - 1
      IPLANE = (J2 - 1) * IROW
C
      DO K = 1, K2
         KK = 1
         IF (K.EQ.K2) KK = 2
C
         DO J = 1, J2
            JJ = 1
            IF (J.EQ.J2) JJ = 2
C
            DO I = 1, I2
               II = 1
               IF (I.EQ.I2) II = 2
C
               MM = (K - KK) * IPLANE + (J - JJ) * IROW + I
     &            - II + 1 + NSTART
               IPOS = IPICK(II, JJ, KK)
C
               Output results for the IPOS'th node of
                  element MM.
C
            END DO
         END DO
      END DO
```

The subroutine given in Appendix C uses this streamlined procedure.

Visualization Techniques

It turns out that programs such as FAST written for displaying results of CFD computations using structured grids have capabilities very useful in displaying seepage/groundwater results. One of the goals of this work

is to apply aerospace engineering technology to the flow of groundwater and seepage under dams. This has proven to be a very successful endeavor. Specific tools and their use will now be described with actual examples given in Chapter VI.

Initial and Final Grid

Of course, the first thing that is needed is the display of the generated grid. Different surfaces of the grid are selected for viewing with various options available, including grid lines, continuous tone shading, and translucent shading. For unconfined flow problems the grid is modified to the shape of the free surface, and the part of the grid where there is no longer any water is collapsed to the free surface. Thus, it is extremely helpful to view the final shape of the grid, as well as the original grid, to visualize the quality of the solution.

Isolevels

One important aspect of the flow net for 2-D applications, such as the one given in Figure 12, is equipotential lines. The 3-D version of an equipotential line is a surface in 3-D space where the potential is a constant value. An isolevel, typically used to look at where a particular value of pressure exists around a structure such as an airfoil, is simply ideal for this alternate use. Different isolevels in a seepage or groundwater problem can be readily analyzed by the practicing engineer to check the

reasonableness of the solution. Incorrect boundary conditions, for instance, will become very apparent.

Color Contours on a Surface

An alternative to the isolevels is color contours on a given surface. The surface can be either one of the i-j, j-k, or k-i planes of the structured grid or an arbitrary cut through the grid. Successive planes can also be viewed to see the progression of results as well.

Particle Traces

The other aspect of the flow net given in Figure 13 is the flow lines. While it is not possible in a general 3-D problem to draw a flow net, the particle traces typically used by the aerospace engineer to trace air flow to find, for instance, vortices are ideal to show the flow of water through the soil. The current version of FAST does a steady-state computation of a particle trace which works very well in this application.

CHAPTER VI

3-D GROUNDWATER FLOW IN AN AQUIFER

Introduction

This chapter contains a three-dimensional problem demonstrating the application of the seepage/groundwater model described in Chapter V. The problem that is given is 3-D groundwater flow in an aquifer. EAGLE is used to generate the grids, the new 3-D seepage/groundwater model is used to do the computations, and FAST is used to display the results. Simpler versions of the problem are first done to accomplish the following:

1. Compare with known results.
2. Investigate the quality of the selected grid.
3. Illustrate the scientific visualization capabilities of FAST.

Of specific importance, also, is whether the numerics of the unconfined portion of the implemented algorithms work properly.

Groundwater Flow in an Aquifer

The problem of groundwater flow in an aquifer will now be presented.

General Description of Problem

The problem of groundwater flow in an aquifer involves flow of water subject to pumping and recharge. Pumping occurs through wells with varying yields. Natural recharge occurs by filtration through the bed of rivers crossing the aquifer area and by seepage through zones of limited extent located at the boundary of the aquifer. Artificial recharge occurs by filtration from the ground surface. Finally, the aquifer is characterized by zones with significantly different material types with some having, for instance, a strong anisotrophy of

$$\frac{k_H}{k_V} > 1$$

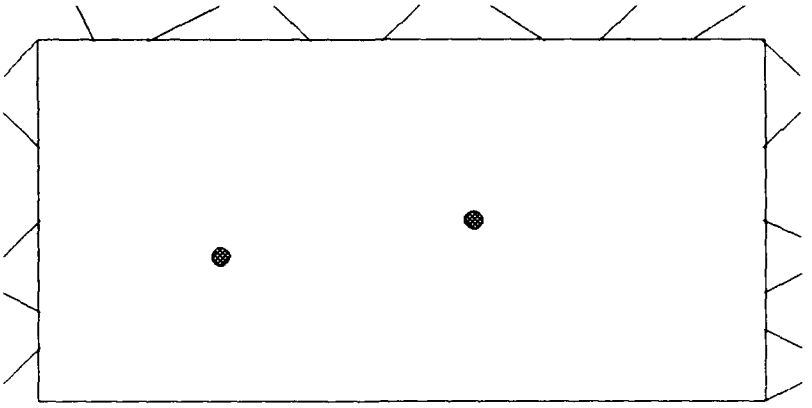
Simplified Problem

The simplified problem will now be discussed and illustrated.

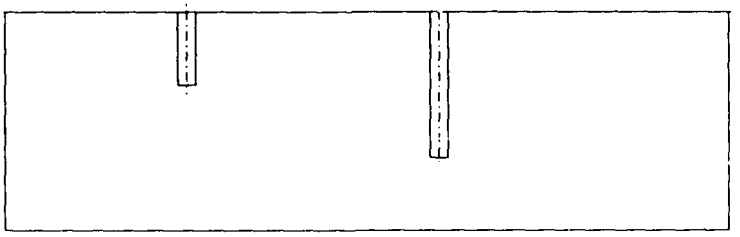
Description of Simplified Problem

The simplified problem is shown in Figure 48 and consists of a very small homogeneous aquifer 1,000 ft long, 500 ft wide, and 120 ft deep. Two partially penetrating wells with the following characteristics have been placed in the aquifer:

Well	X	Y	Radius	Penetration	Q
1	250'	200'	1'	40'	100 cfm
2	600'	250'	2'	80'	200 cfm



a. Plan View



b. Front View

Figure 48. Aquifer with Two Wells

Only confined flow exists in the aquifer, and it is impervious on three sides, as well as the top and bottom. The remaining side, however, is recharged by a river modeled by a constant head of 50 ft above the top of the aquifer. The wells are modeled by the flows, Q_1 and Q_2, being distributed uniformly on the sides of the respective wells. The permeability of the soil is 0.1 ft/min.

Analytic Solution for a Partially Penetrating Well

An analytic solution to a partially penetrating well (Figure 49) of thickness t_w, penetration b, well radius r_w, permeability k, and flow Q has been developed (Muskat 1946), and it will now be given. First, define the variables

$$\left\{ \begin{array}{l} \alpha = \dfrac{r}{2t_w} \\ \zeta = -\dfrac{z}{2t_w} \\ \beta = \dfrac{b}{2t_w} \\ \alpha_w = \dfrac{r_w}{2t_w} \end{array} \right\} \tag{6.1}$$

where (r, z) = (0, 0) is located at the top of the aquifer and at the center of the well. Next, define the function

$$Z(s, y) = \sum_{n=0}^{\infty} \frac{1}{(n + y)^s} \tag{6.2}$$

For small values of α,

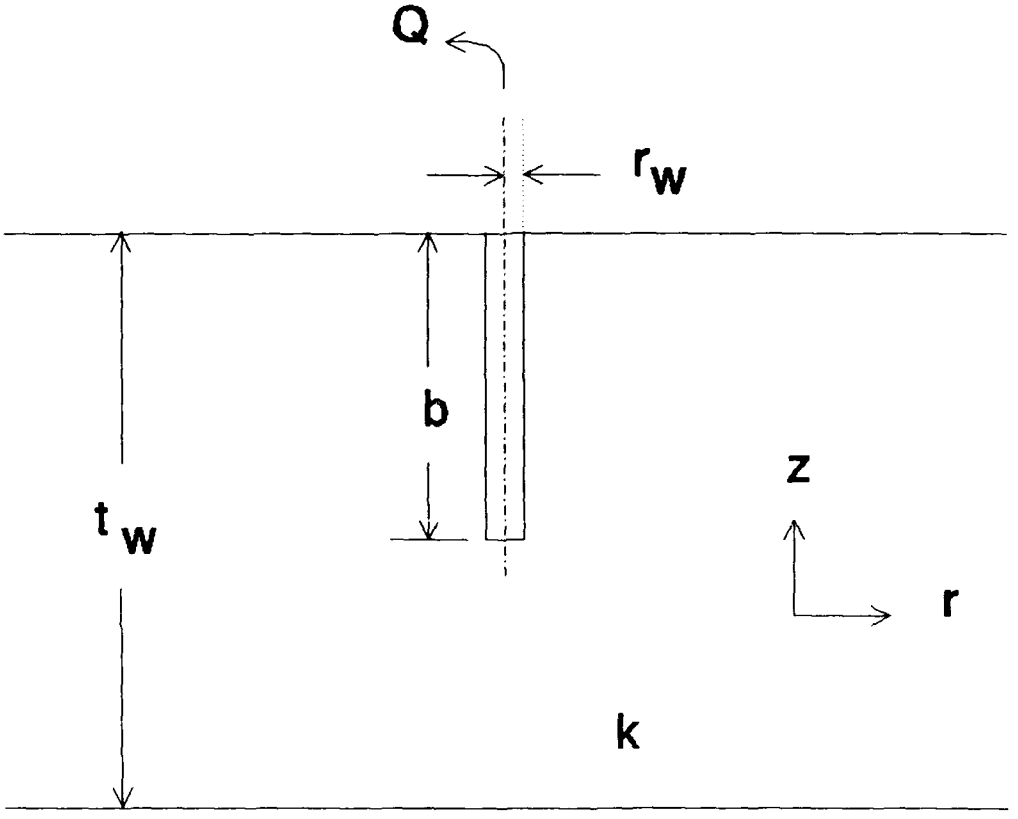


Figure 49. Partially Penetrating Well

$$0 \leq \alpha \leq 0.5$$

the total head, ϕ, becomes

$$\phi = q\{ - \log\frac{\Gamma(1 + \zeta + \beta)\Gamma(1 - \zeta + \beta)}{\Gamma(1 - \zeta - \beta)\Gamma(1 + \zeta - \beta)}$$

$$+ \log\frac{\zeta + \beta + \sqrt{\alpha^2 + (\zeta + \beta)^2}}{\zeta - \beta + \sqrt{\alpha^2 + (\zeta - \beta)^2}}$$

$$- \frac{\alpha^2}{4}[Z(2, 1 - \zeta - \beta) - Z(2, 1 - \zeta + \beta) \qquad (6.3)$$

$$+ Z(2, 1 + \zeta - \beta) - Z(2, 1 + \zeta + \beta)]$$

$$+ O(\alpha^4)\}$$

where the Gamma Function, $\Gamma(x)$, is defined by

$$\Gamma(x) = \int_0^\infty t^{x-1} e^t \, dt$$

q is a flux density given by

$$q = - \frac{Q}{4\pi kb}$$

For the remaining larger values of ρ

$$\phi = 4q \left[\frac{1}{\pi}\sum_{n=1}^{\infty} \frac{i}{n} K_0(2n\pi\alpha) \cos(2n\pi\zeta) \sin(2n\pi\beta)\right. \qquad (6.4)$$

$$\left. + \beta \log\frac{2}{\alpha}\right]$$

where $K_0(x)$ is the modified Bessel Function of the second kind. For relatively large values of x (Press, Flannery, Teukolsky, and Vetterling 1989),

$$K_0(x) \approx \frac{\pi}{\sqrt{2\pi x}} e^{-x}$$

Thus, an implementation of Equation 6.4 requires relatively few terms.

Well in an Aquifer

The equation for the potential of a well in the simplified aquifer shown in Figure 48 can be determined from the solution of a single well by the method of images (Told 1959). Figure 50 illustrates how it works. The original +Q

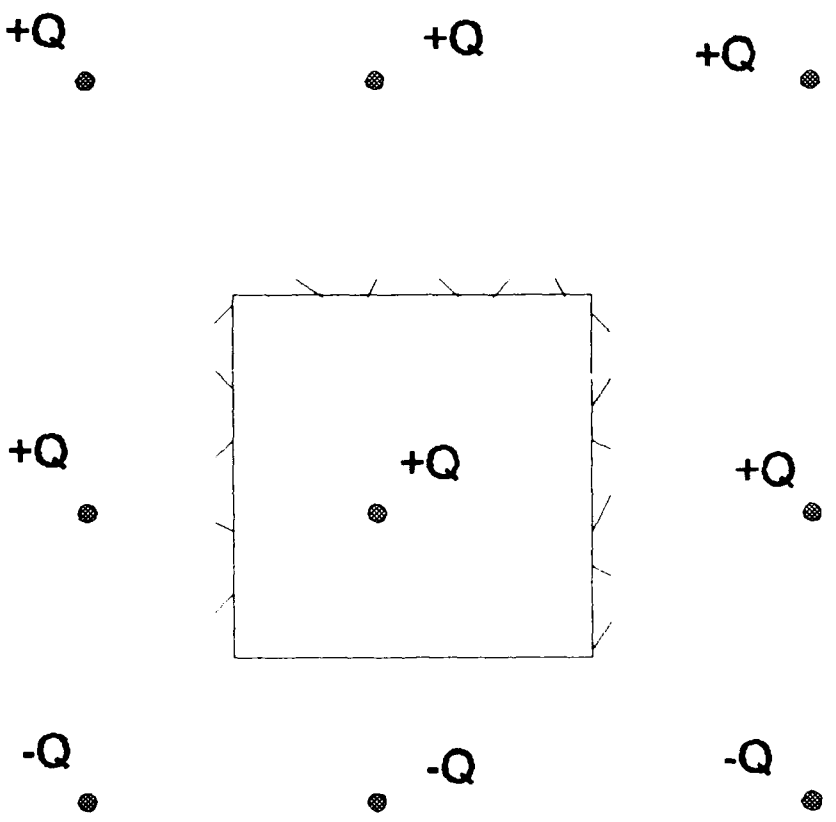


Figure 50. Method of Images

well can have zero potential along the river boundary by adding a -Q image well reflected across the boundary. In like manner, a +Q well can have the impervious boundary condition preserved by adding a +Q image well across the boundary. After the original four image wells have been added, however, additional ones must be added to balance the image wells. In fact, the exact solution is an infinite number of image wells. However, because the radius of influence of a well is from 500 to 1,000 ft, a first or second order approximation is typically all that is required. However, the small problem currently being addressed requires 29 image wells for each original well. This gives 15 wells on either side of the river boundary to provide zero potential at the river.

Equations 6.3 and 6.4 can now be appropriately applied to the original well and the 29 image wells if additional care is taken in defining variables in Equation 6.1. If a well is located at (x_0, y_0) and the top of the aquifer is at z_t, then use

$$\alpha = \frac{1}{2h}\sqrt{(x - x_0)^2 + (y - y_0)^2}$$

$$\zeta = \frac{1}{2h}(z_t - z)$$

The positions and signs of their respective Q's for the 29 image wells are now given. Here, the length of the aquifer is designated by 1, and the origin of the coordinate

system as seen in the plan view is the lower, left-hand corner at the bottom of the aquifer.

Image	X	Y	Sign
1	$-2l + x_0$	$-2t_w - y_0$	+
2	$-x_0$	$-2t_w - y_0$	+
3	x_0	$-2t_w - y_0$	+
4	$2l - x_0$	$-2t_w - y_0$	+
5	$2l + x_0$	$-2t_w - y_0$	+
6	$-2l + x_0$	$-2t_w + y_0$	-
7	$-x_0$	$-2t_w + y_0$	-
8	x_0	$-2t_w + y_0$	-
9	$2l - x_0$	$-2t_w + y_0$	-
10	$2l + x_0$	$-2t_w + y_0$	-
11	$-2l + x_0$	$-y_0$	-
12	$-x_0$	$-y_0$	-
13	x_0	$-y_0$	-
14	$2l - x_0$	$-y_0$	-
15	$2l + x_0$	$-y_0$	-
16	$-2l + x_0$	y_0	+
17	$-x_0$	y_0	+
18	$2l - x_0$	y_0	+
19	$2l + x_0$	y_0	+
20	$-2l + x_0$	$2t_w - y_0$	+
21	$-x_0$	$2t_w - y_0$	+

(Continued)

(Concluded)

Image	X	Y	Sign
22	x_0	$2t_w - y_0$	+
23	$2l - x_0$	$2t_w - y_0$	+
24	$2l + x_0$	$2t_w - y_0$	+
25	$-2l + x_0$	$2t_w + y_0$	-
26	$-x_0$	$2t_w + y_0$	-
27	x_0	$2t_w + y_0$	-
28	$2l - x_0$	$2t_w + y_0$	-
29	$2l + x_0$	$2t_w + y_0$	-

Multiple Wells in an Aquifer

Multiple wells in an aquifer are simply handled by summing the results of the individual wells. Computationally, it is easy to specify a zero potential on the part of the grid modeling the river by selecting a suitable value for the datum. In our simplified problem, for instance, h_0 = 120 ft. The heads will all be zero or negative and represent the drawdown as a result of the pumping of the wells.

Quality of Grids Used

Usually, one can only qualitatively analyze the grids that are used by examining such things as orthogonality, aspect ratio, smoothness, etc., but, in this case, since the exact solution is known, a quantitative comparison can also be made. This offers an excellent opportunity to test some of the different options in EAGLE and different mappings to

see what works best for this problem. One simply computes the percentage error for each grid point for each grid studied. This gives an excellent way of improving the grids used in modeling wells in real-world problems rather than simply using what "looks good."

Solution for a Partially Penetrating Well

First, consider the solution for a partially penetrating well in an infinite aquifer (beyond the well's radius of influence). This will give an indication as to how to do more complex problems using wells as well as show the scientific visualization capabilities of FAST for this application. The problem solved is the second well of the simplified problem. The pertinent data are

$$Q = 200 \text{ cfm}$$
$$k = 0.1 \text{ ft/min}$$
$$b = 80 \text{ ft}$$
$$t_w = 120 \text{ ft}$$
$$\text{Well Radius} = 2 \text{ ft}$$
$$\text{Radius of influence} = 480 \text{ ft}$$

Grid technique

O grid with plug. A grid is constructed first by constructing an O type grid (21 by 21 by 25) as if doing a fully penetrating well (the entire 120 ft) for one block and then for a second block that constitutes a plug (40 ft here with a 6 by 6 by 9 grid) to account for the well being only partially penetrating. Figure 51 shows the O grid representing the soil, and Figure 52 shows the grid for the plug.

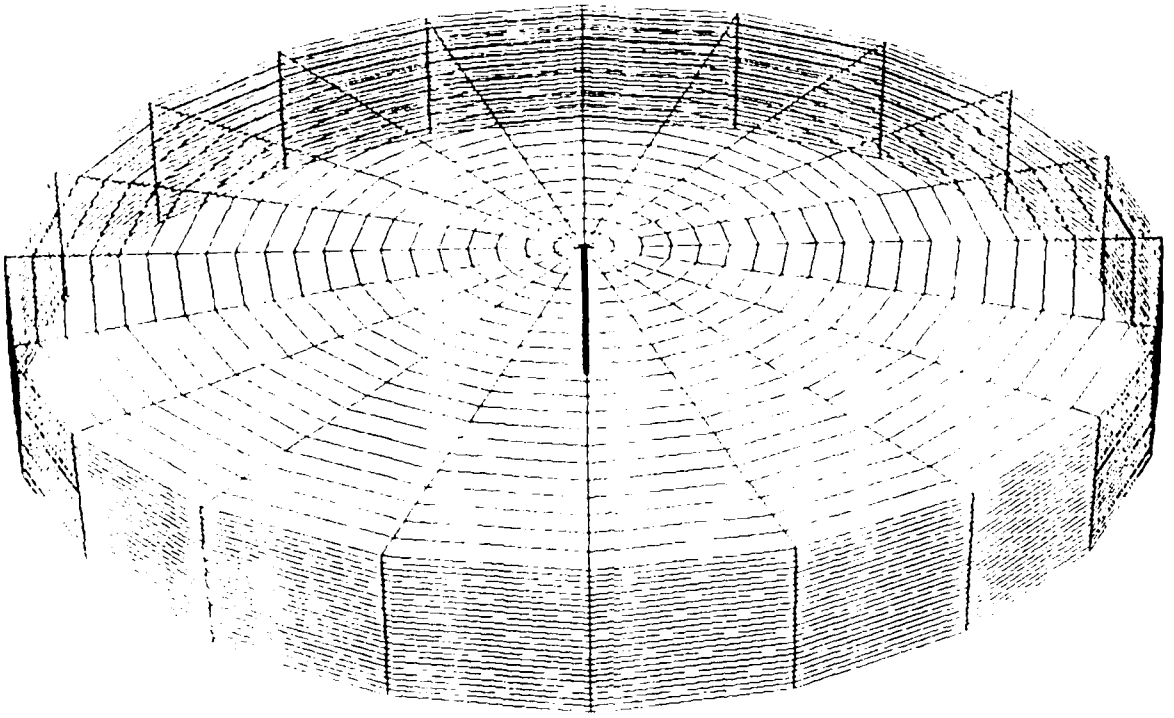


Figure 51. O Type Grid

Note that an O type grid could have been used for the plug, but a different type was chosen.

Spacing. The grids in Figures 51 and 52 use equal spacing. However, it is important to use concentrated spacing toward the well and where the well bottom touches the soil. Therefore, a second grid and solution was obtained using concentrated spacing. Figure 53 shows the radial distribution of nodes, and Figure 54 shows the distribution with depth of this grid.

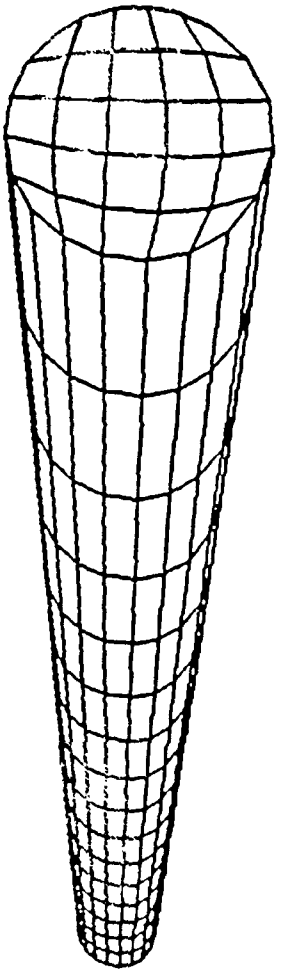


Figure 52. Plug

Analysis of results

Visualization. First, the results were viewed using FAST to consider the quality of the solution. It turns out that FAST works very well for this purpose. Figure 55 shows a color contour plot of total head for a vertical section (constant I) using the module Surfer. This plot is very similar to plotting equipotentials as half of a flow net in

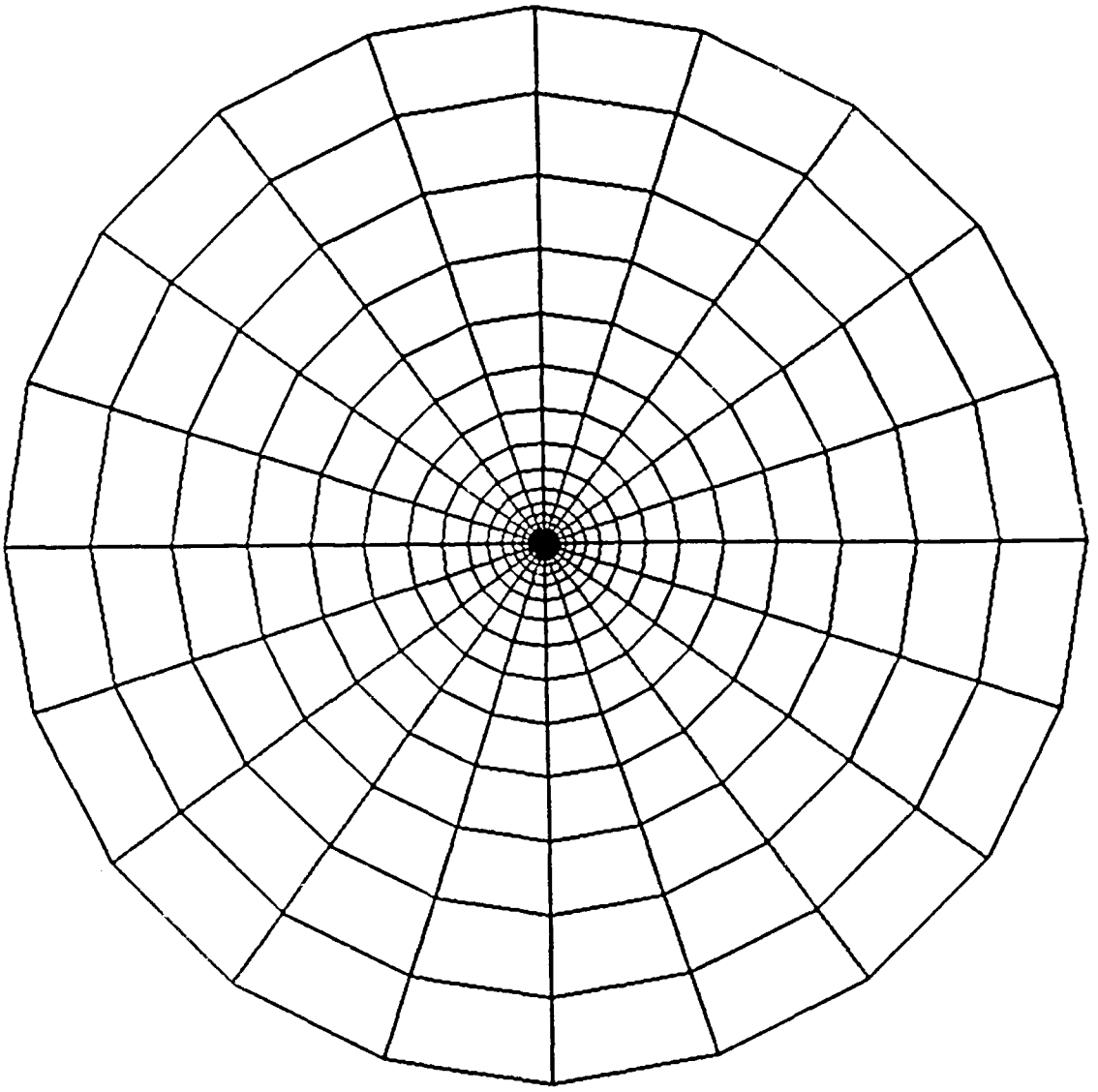


Figure 53. Radial Spacing

2-D applications. The results show very clearly horizontal flow until the water approaches the well. Figure 56 shows some constant J color contour plots. The constant color bands indicate horizontal flow as well. As one approaches the well, the constant J surfaces have different colors showing other than horizontal flow. Figure 57 shows a constant

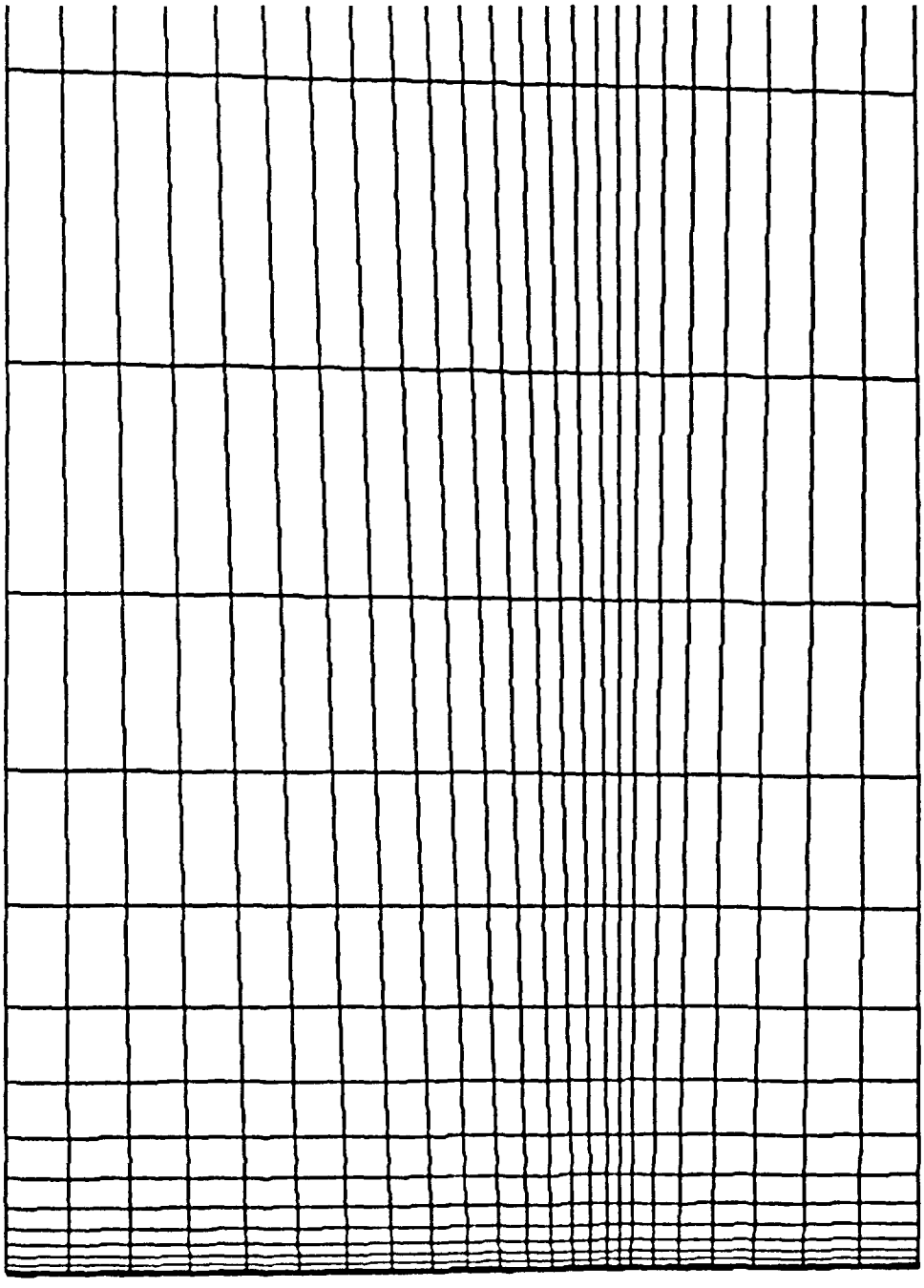


Figure 54. Depth Spacing

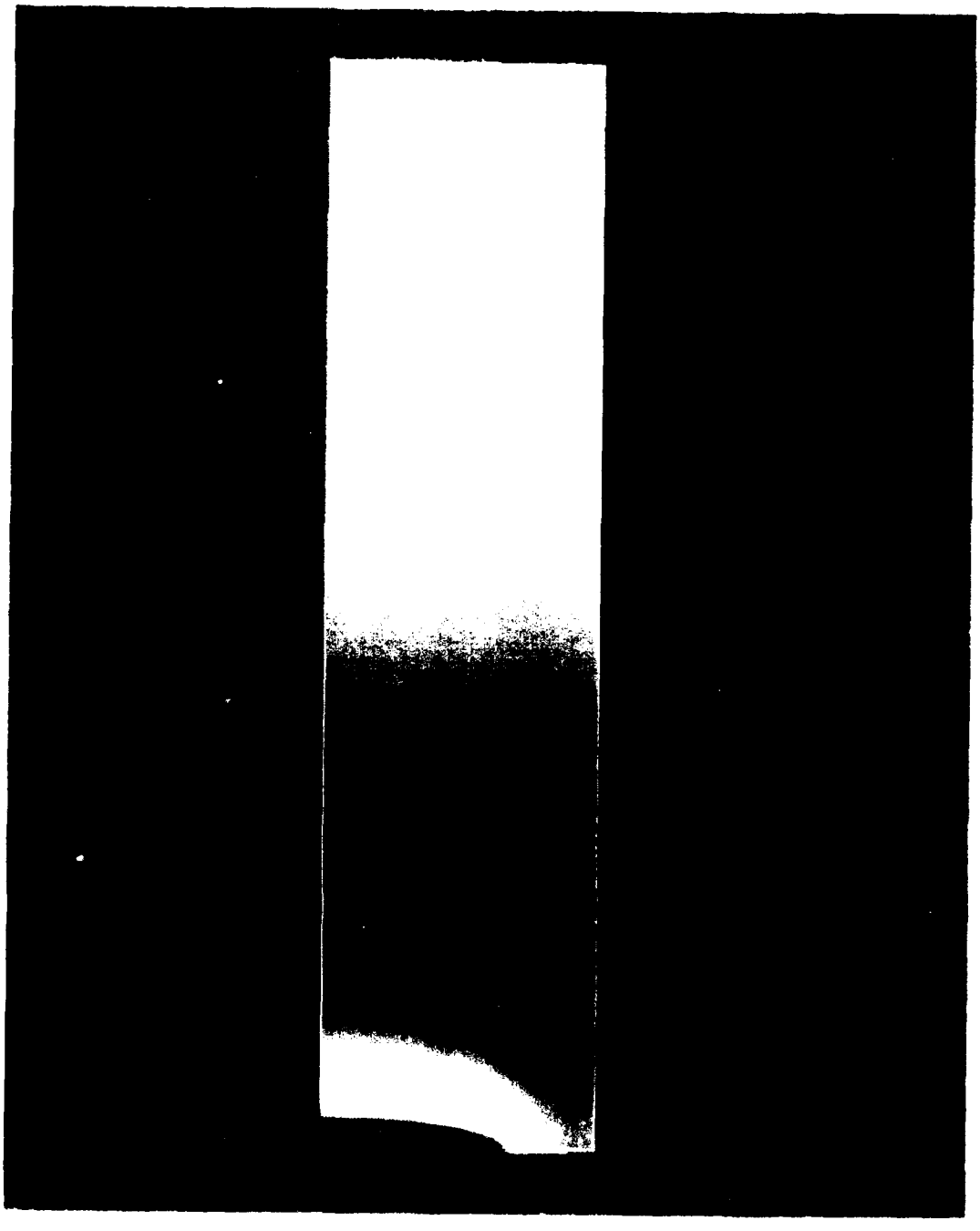


Figure 55. Constant I Color Contour Plot

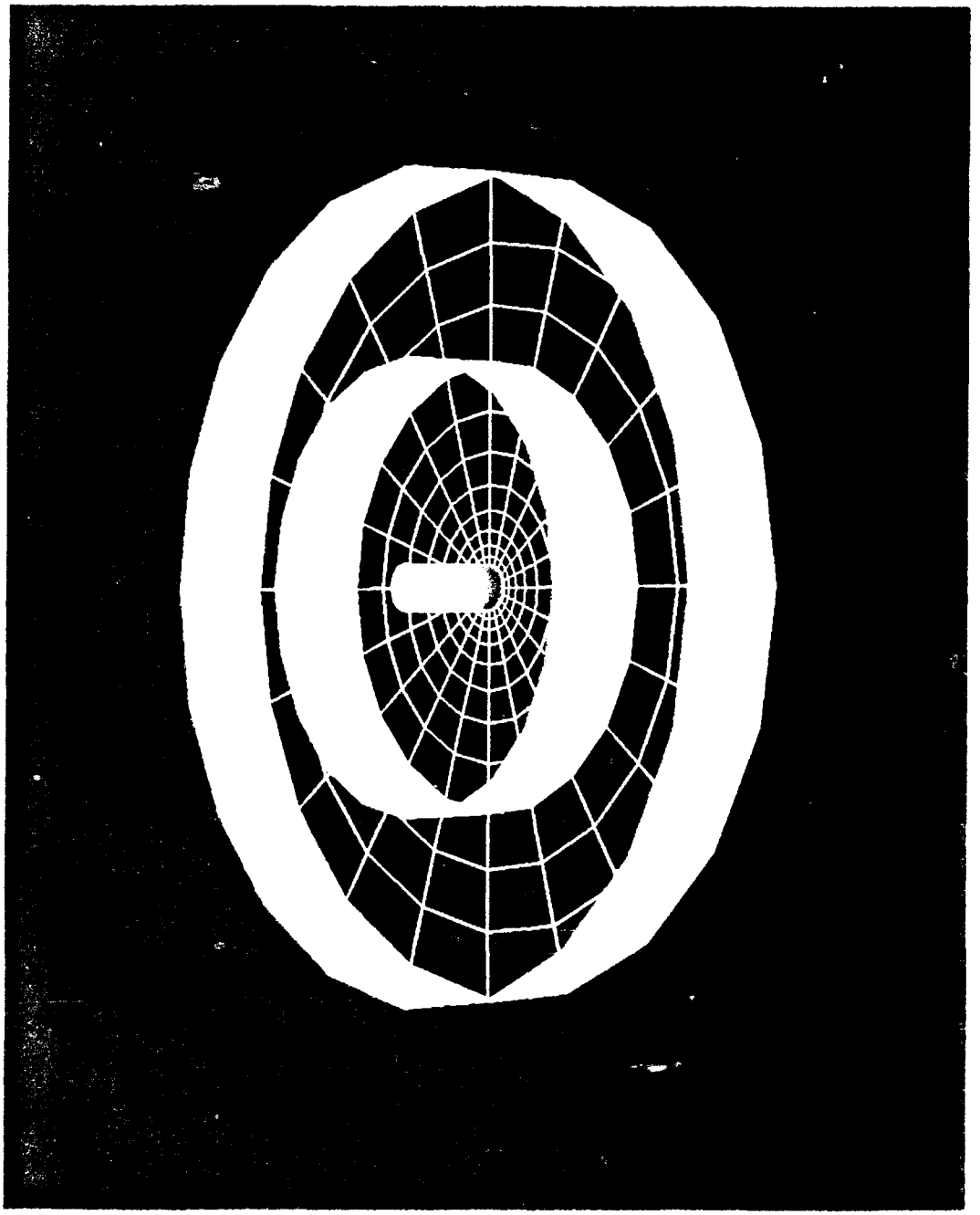


Figure 56. Constant J Color Contour Plot

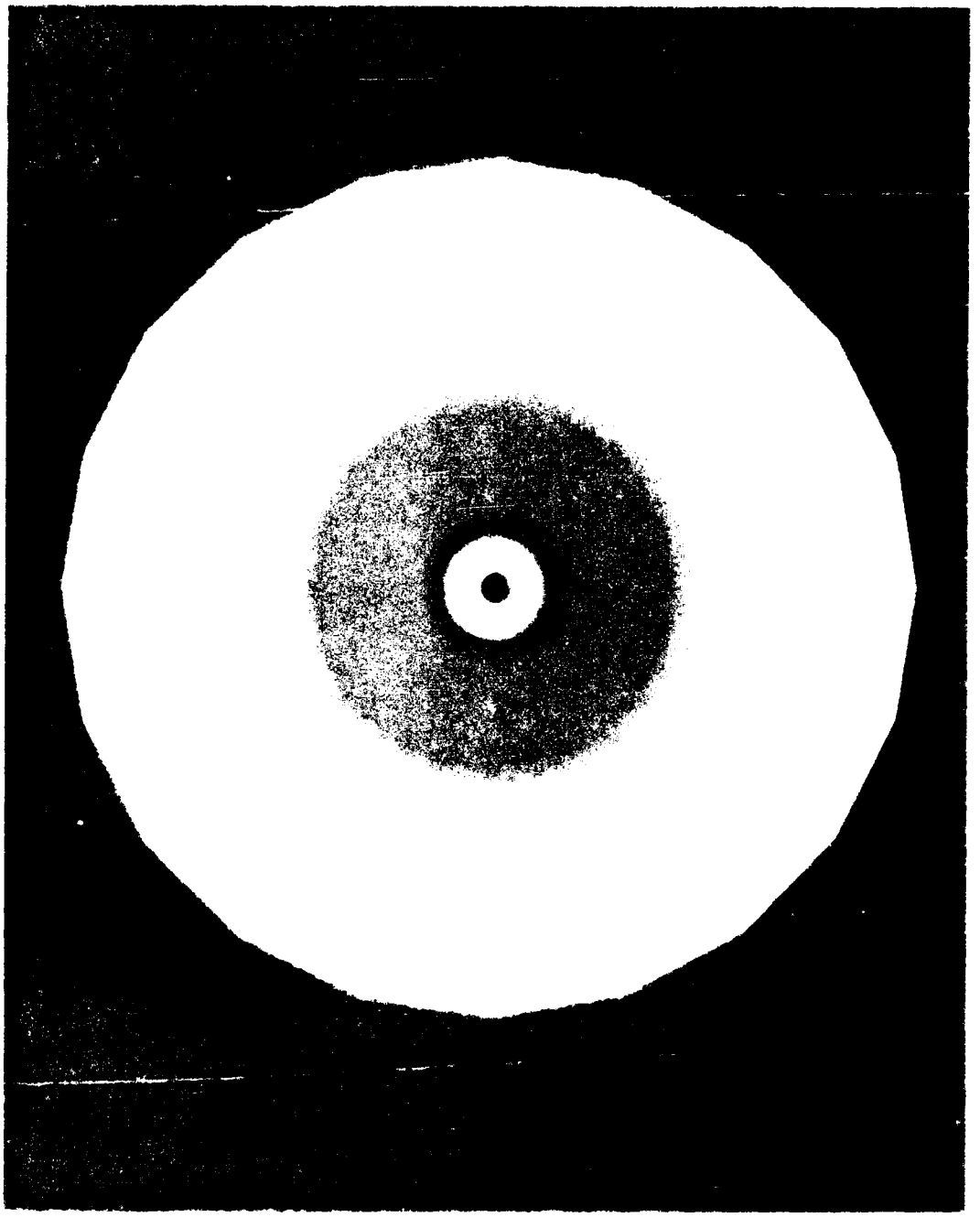


Figure 57. Constant K Color Contour Plot

K color contour plot. First, the problem is axisymmetric, so the solution must be as well, which is the case. Also, the variation is approximately logarithmic as should be.

Another type of visualization technique using module Isolevel is shown in Figure 58. The 2-D problem has equipotential lines, but the 3-D problem has isolevel surfaces. Again, since flow occurs perpendicular to these surfaces, they will be vertical rings until the well is approached.

Figure 59 shows the final type of visualization used (generated from module Tracer). It is a particle trace for a vertical section of the soil near the well. What the aerospace engineers use for showing the flow of air around an aircraft is also an excellent tool for showing flow in porous media. The part of the grid that represents the well is shown in red, and the flow lines are shown in black. Again horizonal flow exists except near the well.

The first value of these visualization tools is to aid in deciding whether the solution appears reasonable. If incorrect boundary conditions or poor answers result, this becomes readily apparent. A second value is once confidence has been obtained in the model, specific details as to what is happening can then be analyzed.

Error analysis. Equations 6.3 and 6.4 were used to compute head or potential for each of the 10,644 nodes, and the results were compared with the computed results. The break point for α between using Equation 6.3 or 6.4 is

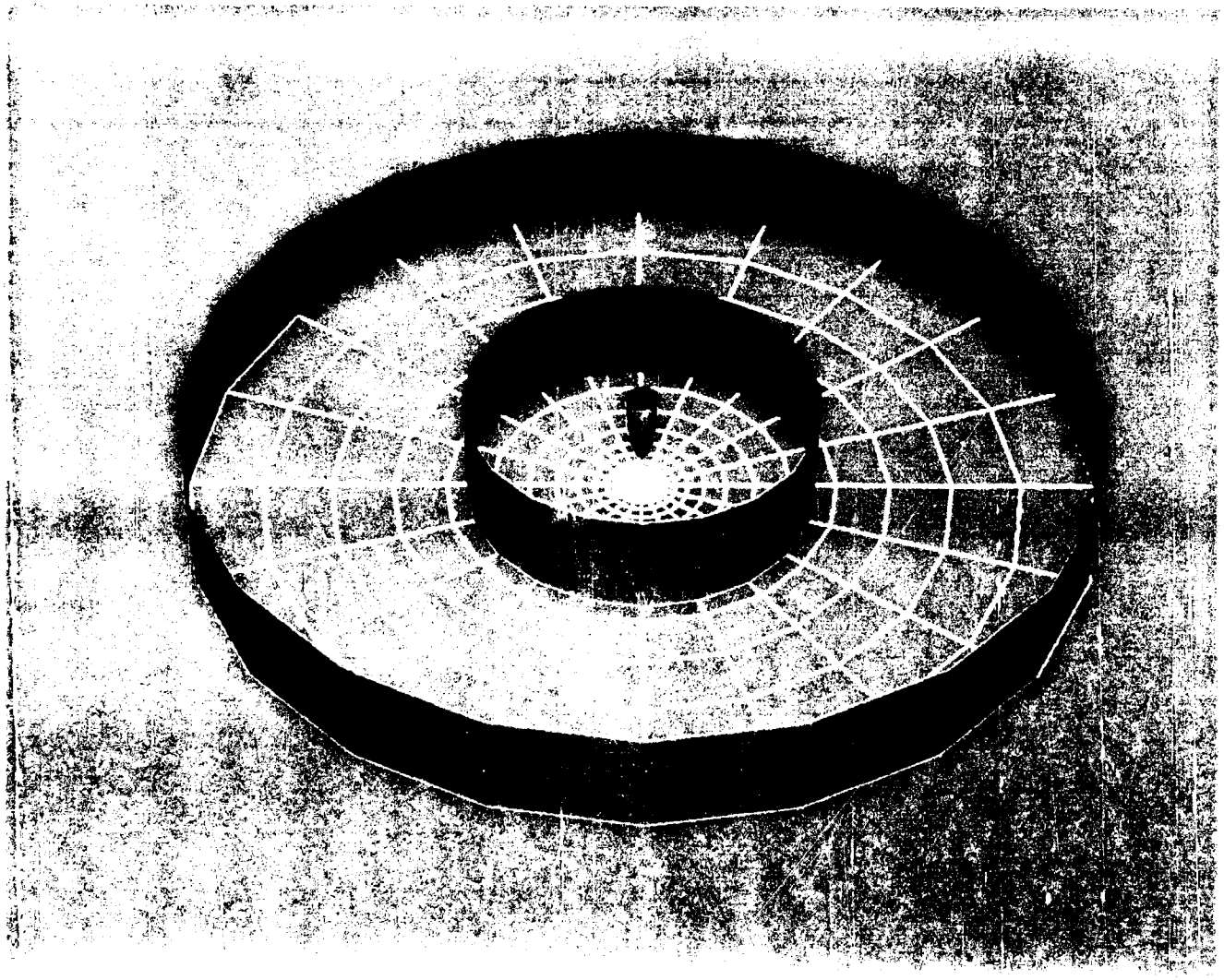


Figure 58. Isolevel Surfaces Plot

Figure 59. Particle Trace Plot

$$\alpha \approx 0.2$$

The grid with equal spacing had a maximum percentage error of 24.1 percent. Since the heads near r = 480 ft approach zero, the percent error was computed by

$$\% \epsilon = \left| \frac{h_c - h_t}{h_w} \right| \times 100\%$$

where

h_c = computed head

h_t = theoretical head

h_w = maximum head at the well

Figure 60 shows a color contour plot of percent error for this grid with white being the worst case. It is remarkable that with even spacing the error is very small until you are very near the well. The default spacing in EAGLE was used, and the maximum percent error for this grid was 4.6 percent.

Solution of Simplified Problem

Now the solution for the problem given in Figure 48 will be given. Because quality of grids is an important issue, different single and multiple block configurations were considered. Because of its potential to real-world problems and the developed capability to merge separate blocks, the selected solution is to enclose each well in an O type box grid (Figure 61) and merge them with their

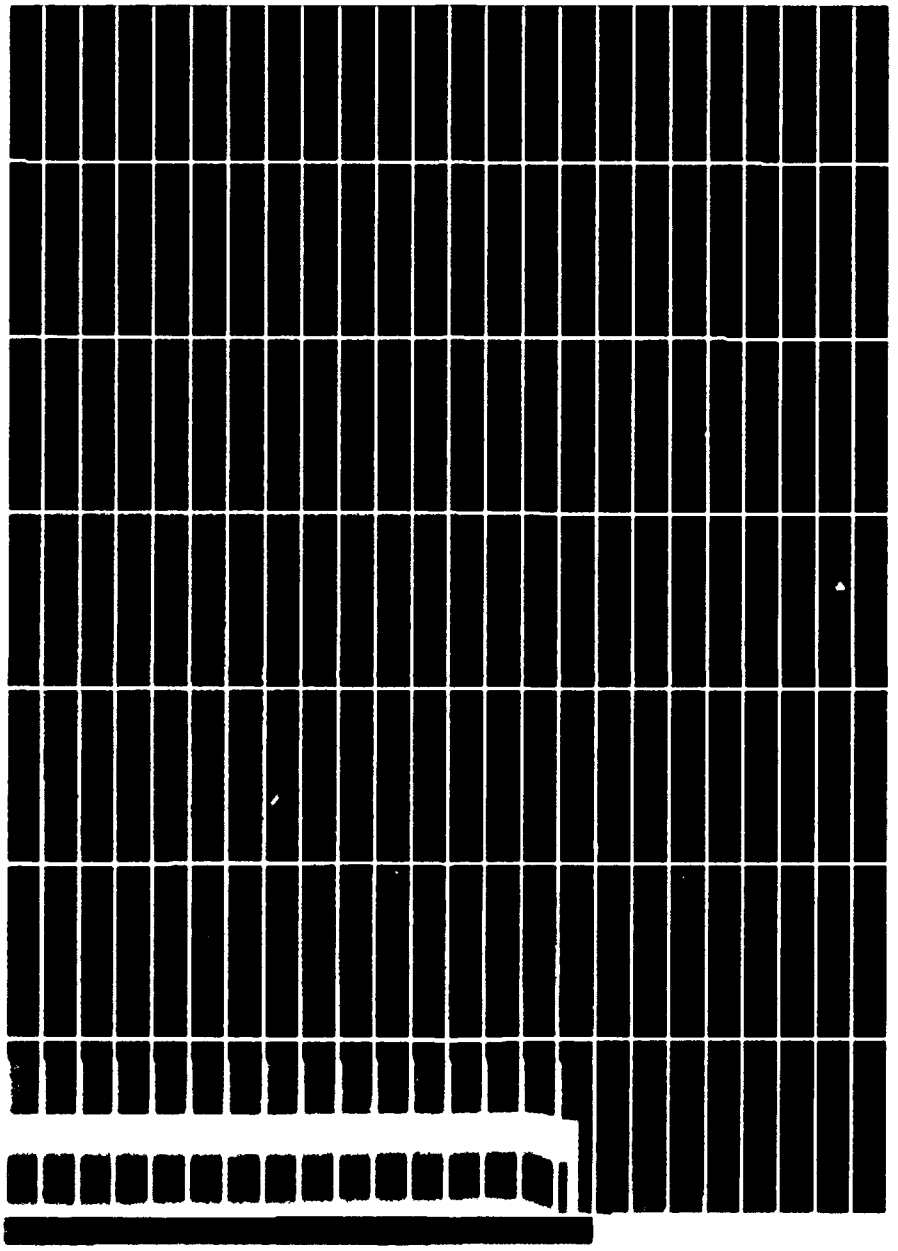


Figure 60. Percent Error Plot

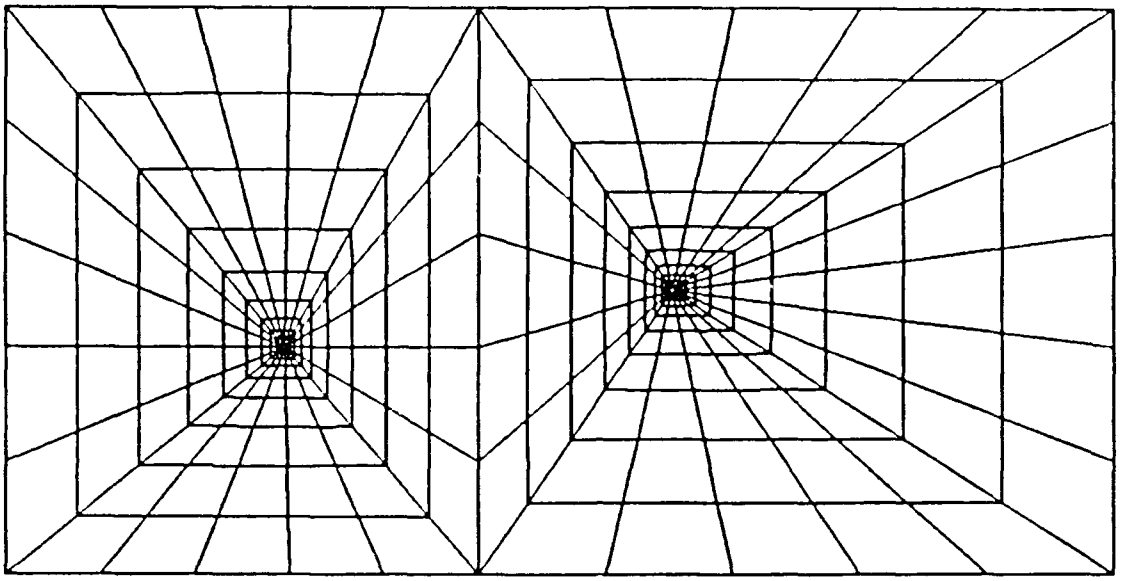


Figure 61. Two-Well Problem

respective plugs and each other. Two versions will be presented here. A large algebraic grid of 21,266 nodes will be first considered and then compared with a much smaller elliptic grid of 13,266 nodes using the contyp='initial' option in EAGLE. Appendix D contains a description of the data required to generate the elliptic FEM grid.

Analysis of results

Visualization. First, the large algebraic grid at the well is shown in Figure 62. The distribution of nodes from the circular well to the rectangular boundary is quite good before smoothing is done. Figures 63 through 65 show color contours of total head for J, I, and K levels, respectively. Here, the highest value (white) is at the river, and the

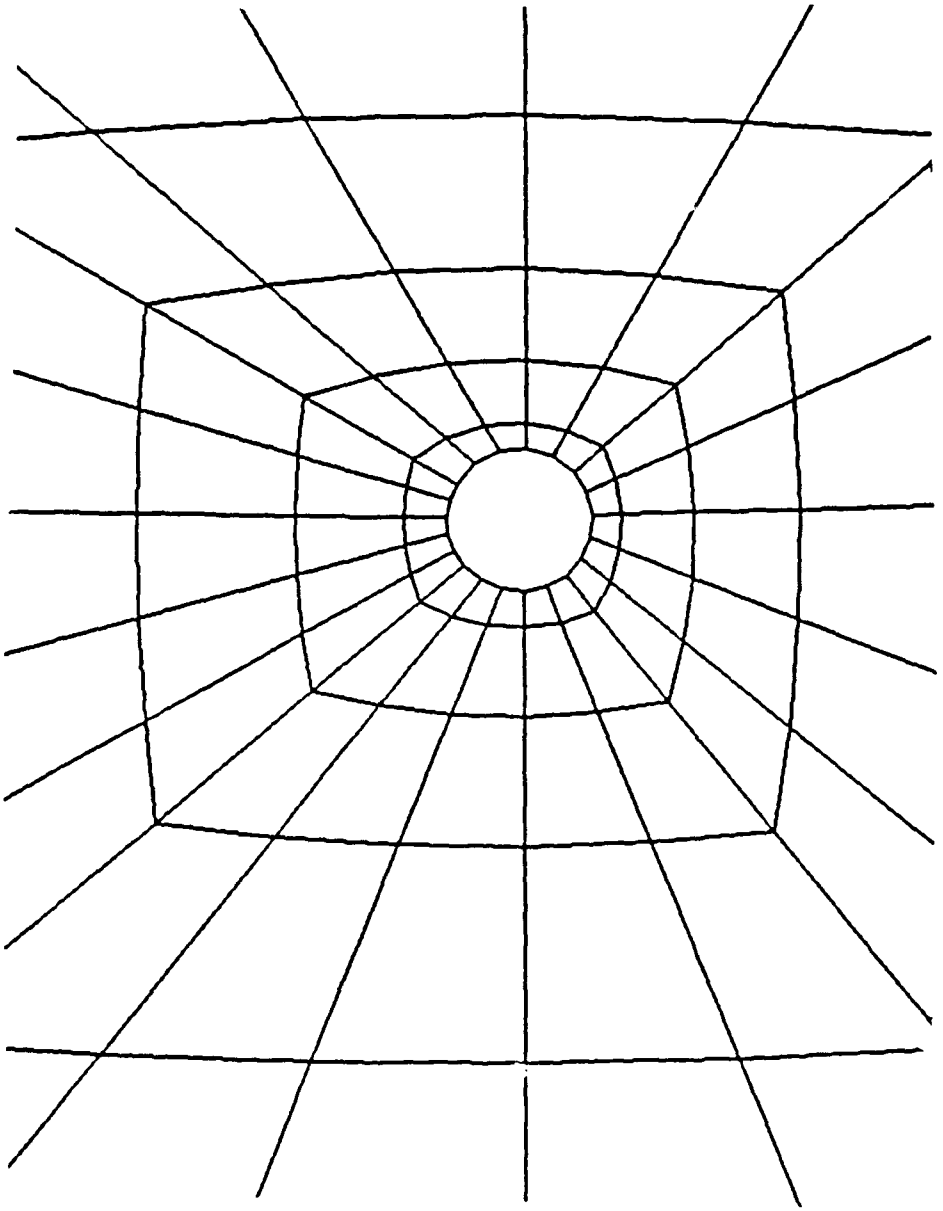


Figure 62. Large Algebraic Grid at Well

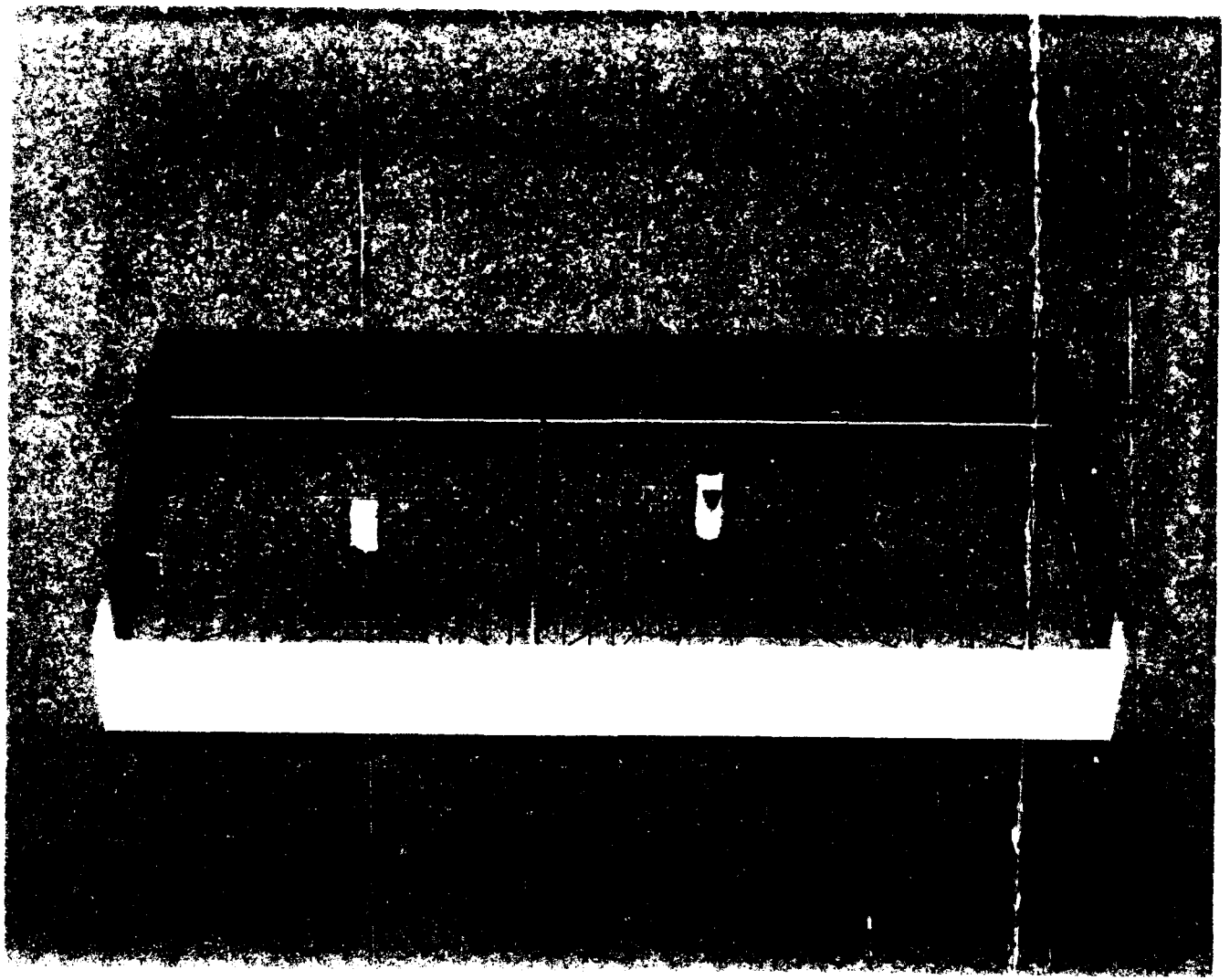


Figure 63. J Surface Color Contour Plot

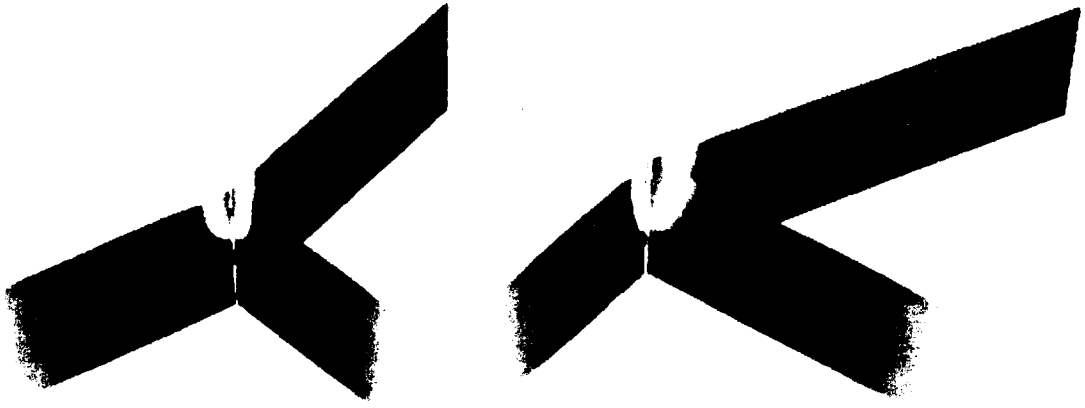


Figure 64. I Surface Color Contour Plot

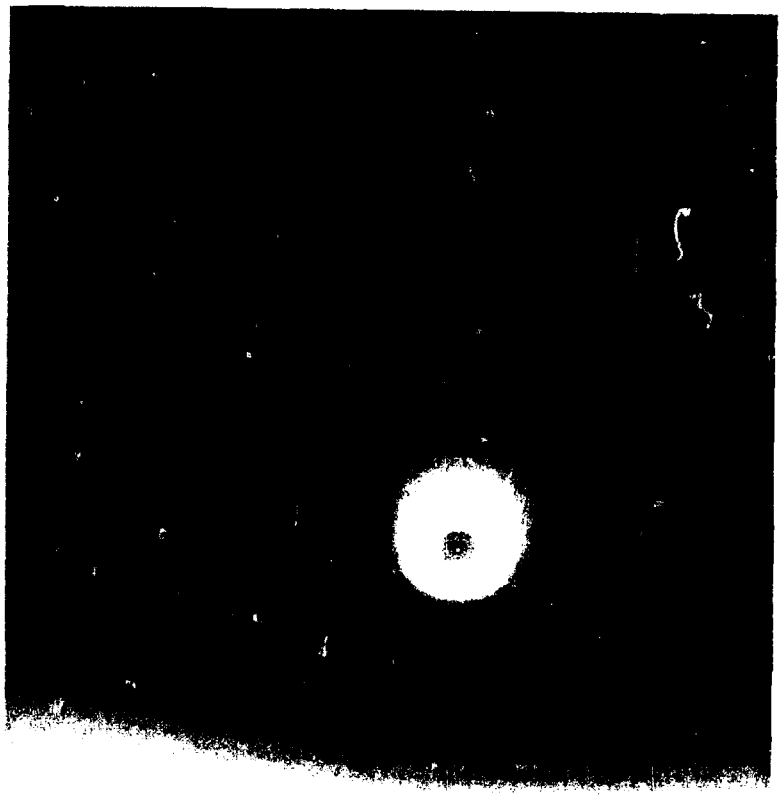


Figure 65. K Surface Color Contour Plot

lowest value of head (black) is at the wells. Figure 66 shows the isolevel plot for the two well system. Figure 67 shows flow lines (particle traces) going into the first well. All flow starts at the river and ends at one of the two wells. One surprising result is that the left most flow line skips the smaller well and goes for the bigger one.

Error analysis. The large algebraic grid of 21,266 nodes created from essentially putting together two single well grids had a maximum percentage error of 5.3 percent. Actually, some of this error is attributed to the truncation in the number of image wells and some is due to numerical imperfections. It seemed plausible that this grid could be significantly reduced in size by applying the elliptic grid generation techniques of EAGLE to get close to the same result. Figure 68 shows a K surface for the first well for the smaller elliptic grid of 13,266 nodes. This grid yielded a maximum percentage error of 5.7 percent which substantiates the hypothesis.

Real-World Example

A real-world example showing the solution process will now be presented.

Description of Problem

The problem shown in plan view in Figure 69 consists of a part of an aquifer containing a river crossing through the region with partially penetrating wells pumping water out of

Figure 66. Isolevel Plot

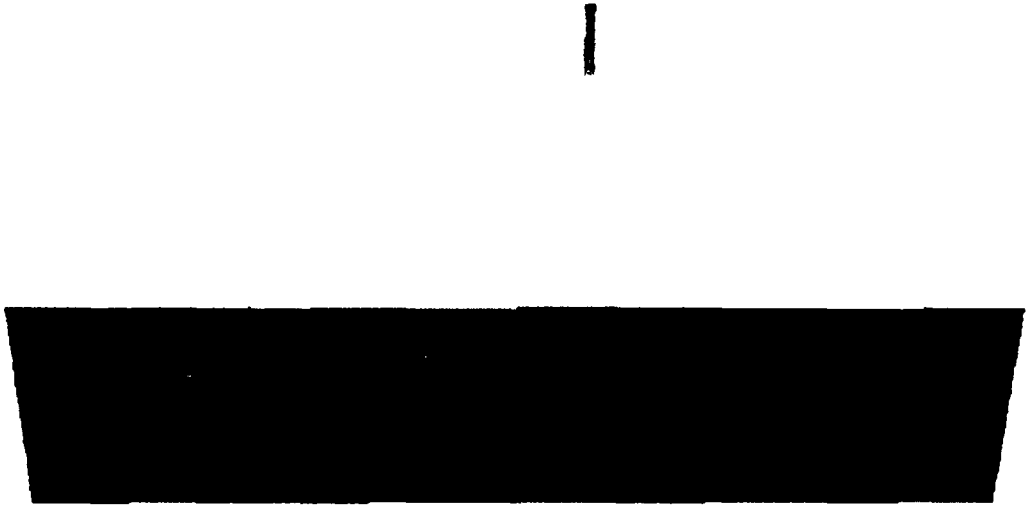


Figure 67. Flow Lines Plot for One Well

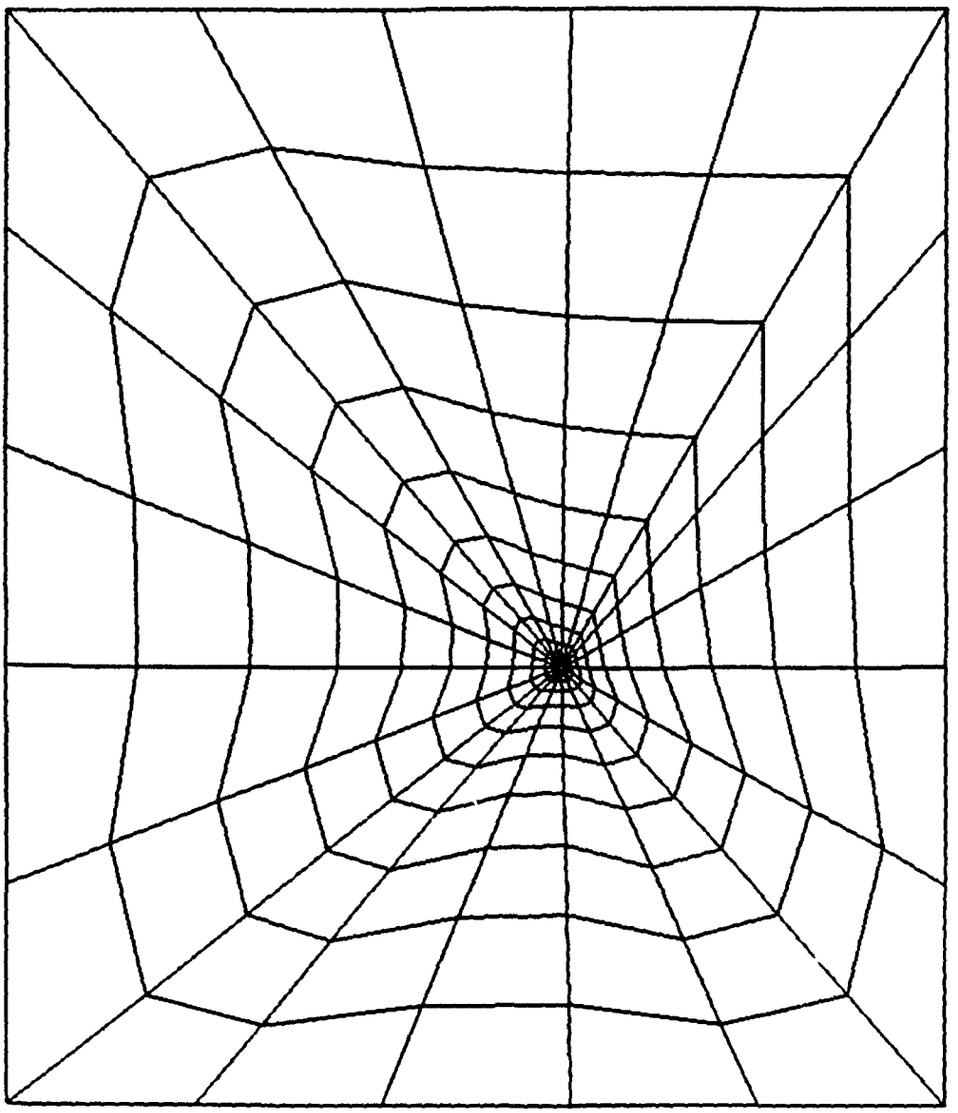


Figure 68. Elliptic Grid

the system. Line segment EF is an impervious wall such as a slurry trench found at certain sensitive military arsenals and construction sites. The region with the two wells is

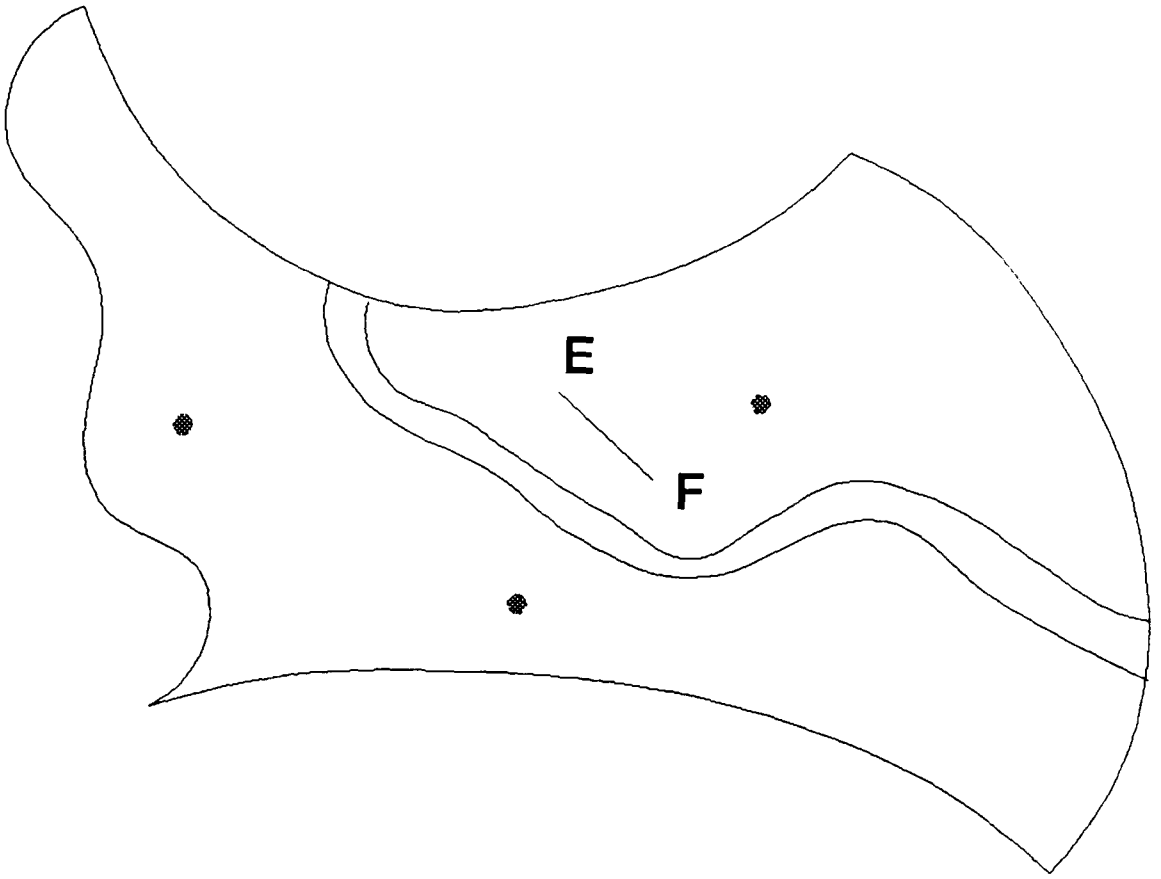


Figure 69. Plan View of Aquifer

highly anisotropic and has the following permeability data (see Figure 70):

$$\theta_e = 21^\circ$$

$$\phi_e = -53^\circ$$

$$\psi_e = 10^\circ$$

$$k_{p1} = 0.01\ ft/\text{min}$$

$$k_{p2} = 0.015\ ft/\text{min}$$

$$k_{p3} = 0.001\ ft/\text{min}$$

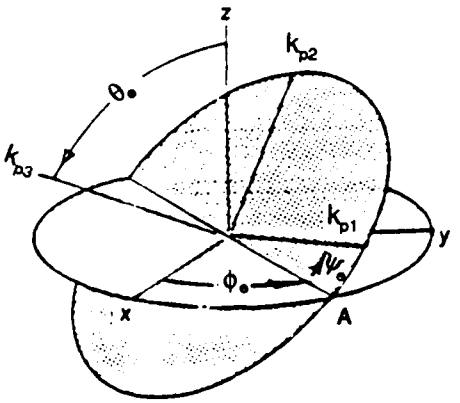


Figure 70. Permeability Orientation

The region under the river is a rather impervious clay having the soil properties:

$$\theta_e = 0^\circ$$

$$\phi_e = 0^\circ$$

$$\psi_e = 0^\circ$$

$$k_{p1} = 0.00001 \ ft/\text{min}$$

$$k_{p2} = 0.00001 \ ft/\text{min}$$

$$k_{p3} = 0.00001 \ ft/\text{min}$$

Finally, the region with the slurry trench has soil properties of a pervious sand:

$$\theta_e = 0^\circ$$

$$\phi_e = 0^\circ$$

$$\psi_e = 0^\circ$$

$$k_{p1} = 0.1\ ft/\text{min}$$

$$k_{p2} = 0.1\ ft/\text{min}$$

$$k_{p3} = 0.1\ ft/\text{min}$$

The thickness of the aquifer varies between approximately 200 to 350 ft.

FEM Grid

This problem provides an excellent opportunity to test the techniques and concepts developed thus far. The region was divided into 16 blocks or subregions as shown in Figure 71. Thirteen subregions are visible with three plugs making partially penetrating wells not shown. Decoupling the geometry makes it much easier to generate the grid at times. It is certainly not necessary to have so many blocks as the mapping capability of EAGLE is extensive. However, it was done this way to give the approach a thorough test. Once the data files for the surf and grid modules of EAGLE were developed for a particular type of region, it was very easy to modify these files for the next similar type region. After all the pieces were generated, it was a simple matter to combine the pieces into a single, complete FEM grid using

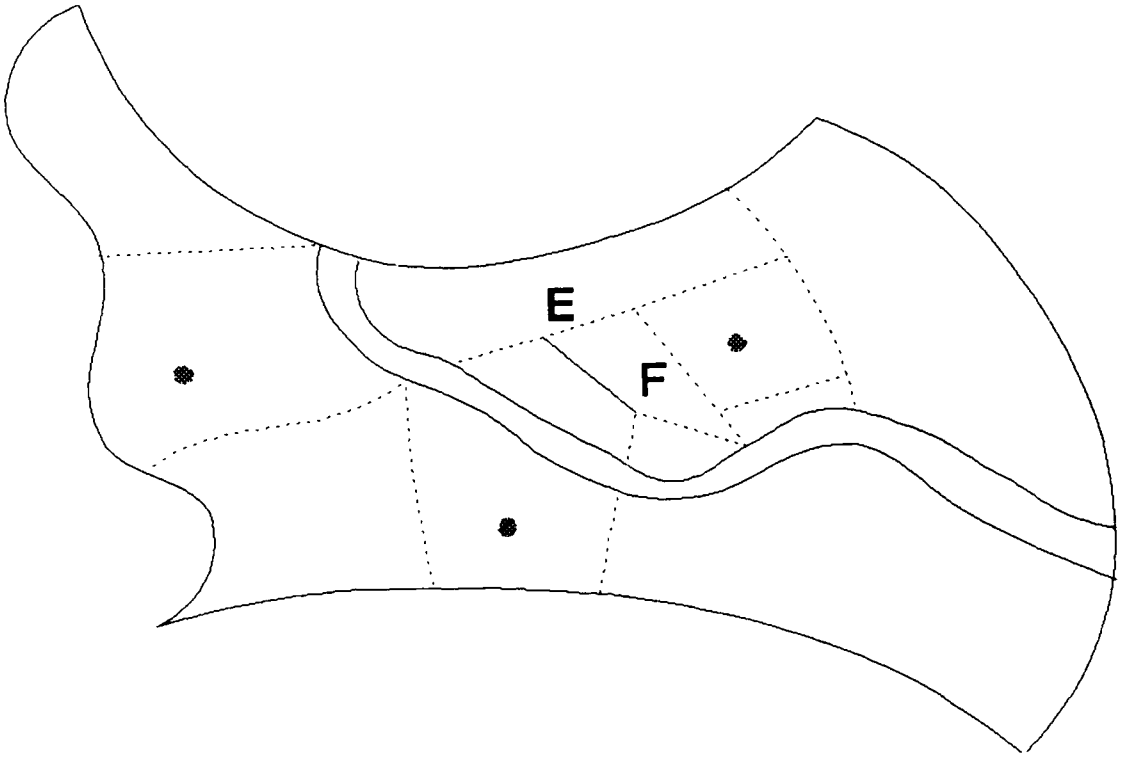


Figure 71. Subregions

the program developed as part of this work. Figure 72 shows a plan view of the grid, and Figure 73 shows a perspective view of the grid and aquifer system. Eleven layers (K levels) were used with the resulting FE grid having 11,578 nodes and 9,855 elements.

With the head at the river being 300 ft, the wells each having a penetration of 100 ft, and a head of 260 ft at the well, the following data was used to generate the FE grid for the problem assuming the 16 pieces have already been generated:

```
com
16
aquif1.eg1
1 21. -53. 10.
aquif2.eg1
1 21. -53. 10.
aquif3.eg1
1 21. -53. 10.
aquif4.eg1
1 21. -53. 10.
aquif5.eg1
1 21. -53. 10.
aquif6.eg1
2 0 0 0
aquif7.eg1
3 0 0 0
aquif8.eg1
3 0 0 0
aquif9.eg1
3 0 0 0
aquif10.eg1
3 0 0 0
aquif11.eg1
3 0 0 0
aquif12.eg1
3 0 0 0
aquif13.eg1
3 0 0 0
aquif14.eg1
1 21. -53. 10.
aquif15.eg1
1 21. -53. 10.
aquif16.eg1
3 0 0 0
ban
bc
2 1 1 6 23 1 8 1 260.
bc
2 1 1 9 23 1 9 12 260.
bc
2 1 1 10 23 1 11 2 0.
bc
5 1 1 6 23 1 8 1 260.
bc
5 1 1 9 23 1 9 12 260.
bc
5 1 1 10 23 1 11 2 0.
bc
8 1 1 6 21 1 8 1 260.
bc
8 1 1 9 21 1 9 12 260.
```

```
bc
8 1 1 10 21 1 11 2 0.
bc
6 1 1 11 23 5 11 1 300
out
aquifer
Aquifer with Unconfined Flow
0.
.01 .015 .001
.0001 .0001 .0001
.1 .1 .1
end
```

The grid around each well was generated by going directly from a circle to a rectangular type region with care to use gradually increasing spacing (Figure 74). For some applications there may be too much skewness, and an alternate plan such as a five-region system must be implemented. However, as will be shown later, good results were achieved in this application using the original approach.

The impervious wall can be easily handled as shown in Figure 75. All that must be done is to have nodes at different positions on the sides of the wall.

Presentation of Results

The problem was first run as having confined flow and a homogeneous, isotropic medium. Figure 76 shows a color contour plot of potential for the top value of k = 11 with white being full potential and black representing the lowest value of potential. An exception is the river which is at full potential but is painted blue for visualization effectiveness. Radial flow at the well is seen which is to be

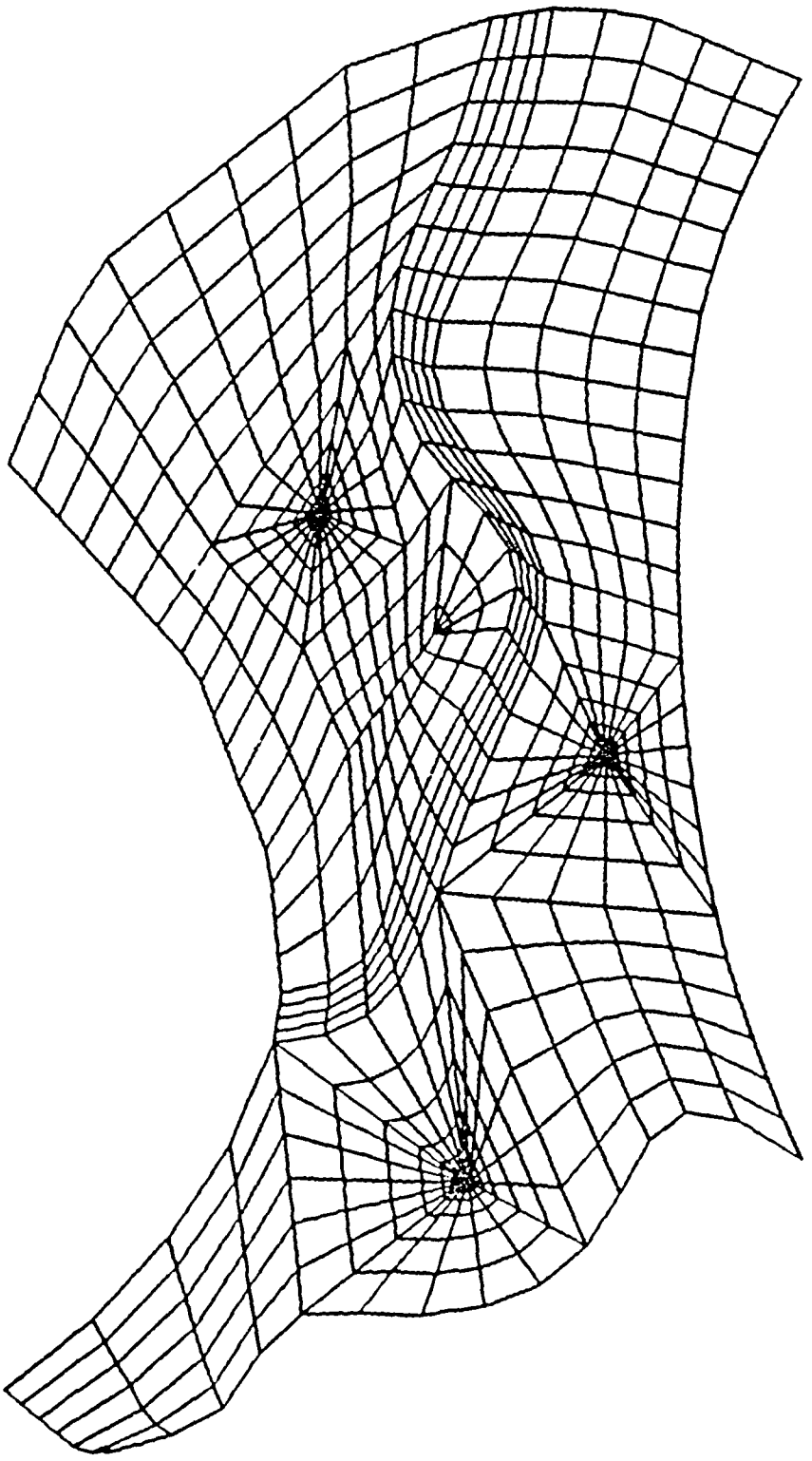


Figure 72. Plan View

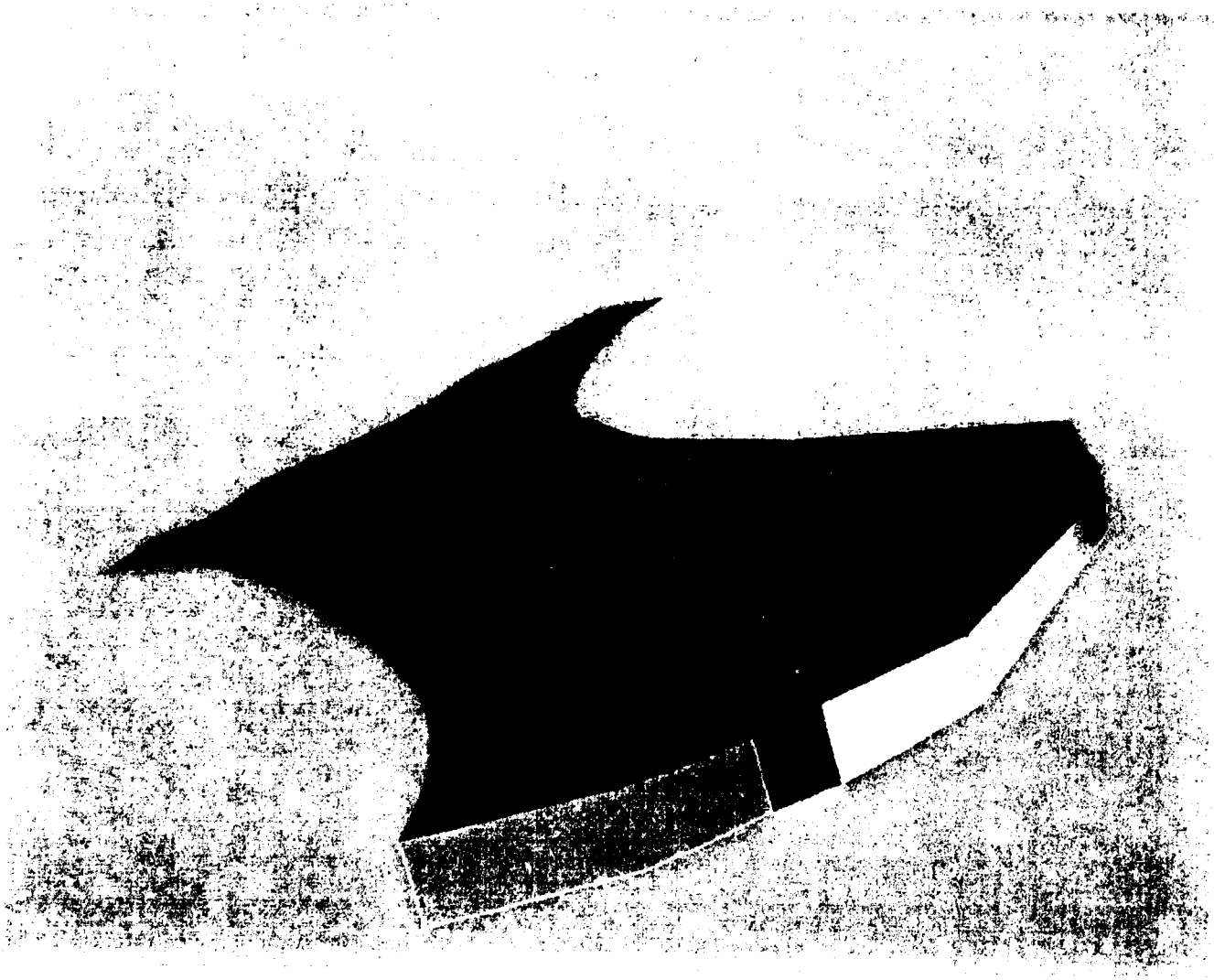


Figure 73. Perspective View

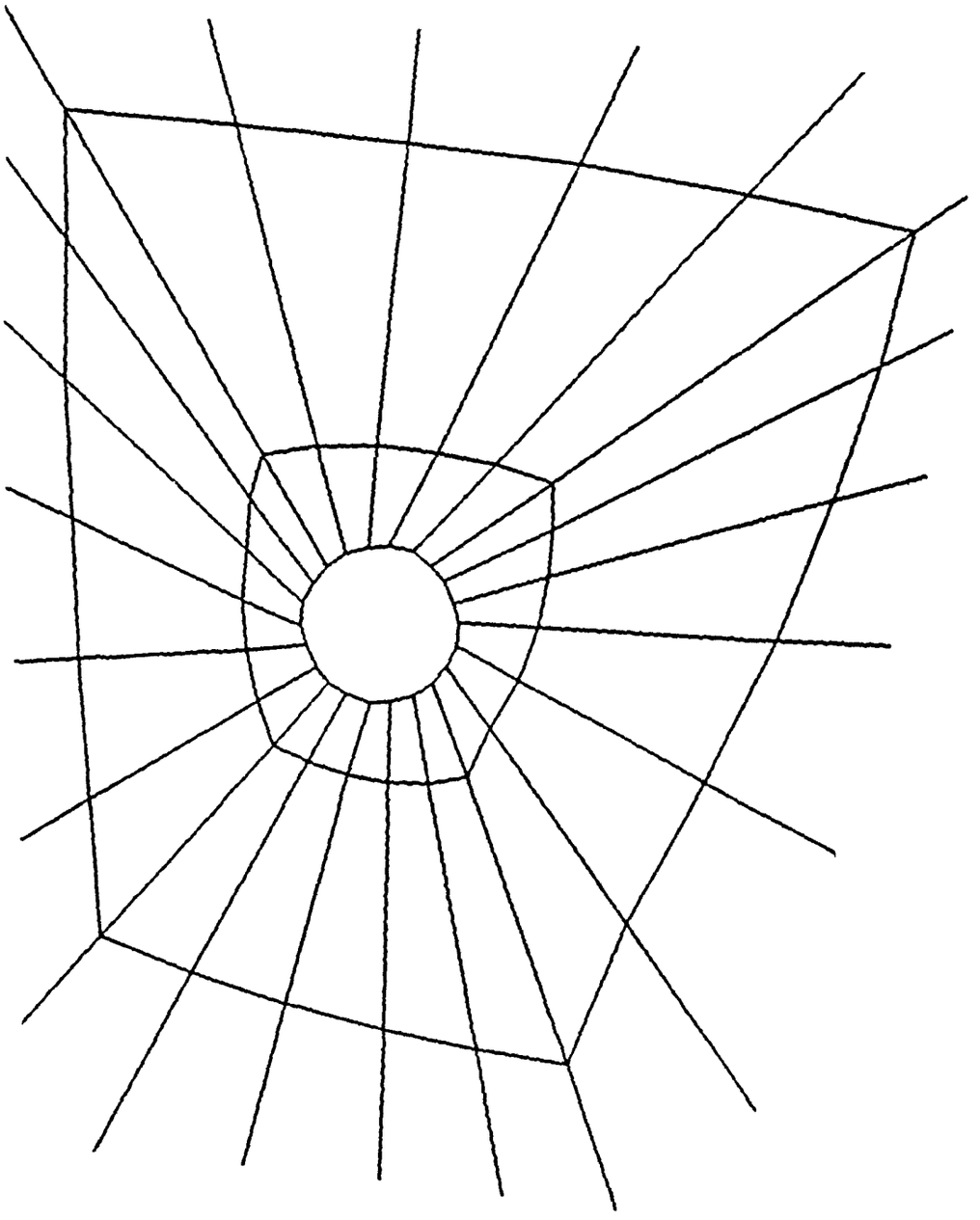


Figure 74. Grid at Well

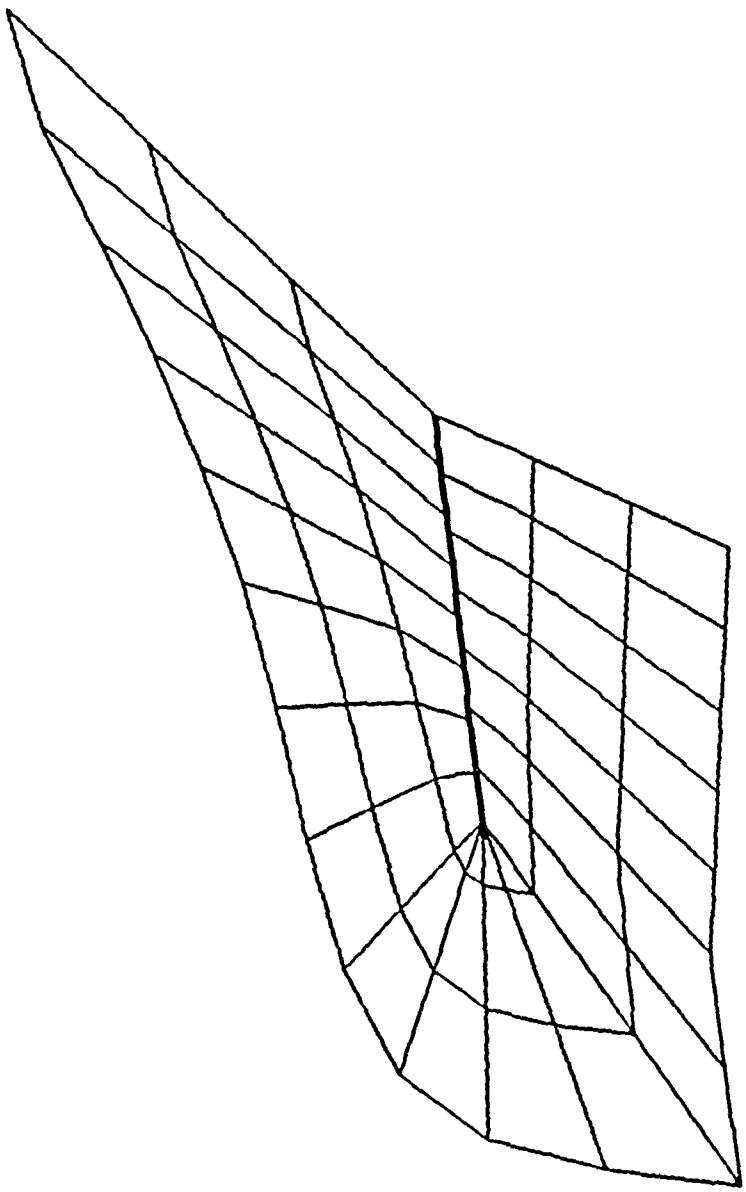


Figure 75. Grid at Impervious Wall

expected. It is also interesting to see the effect of the impervious wall on the head distribution.

The unconfined flow problem was then run for the real soil conditions, requiring a running time of 3763.2 sec on the Cray YMP and five iterations for convergence.

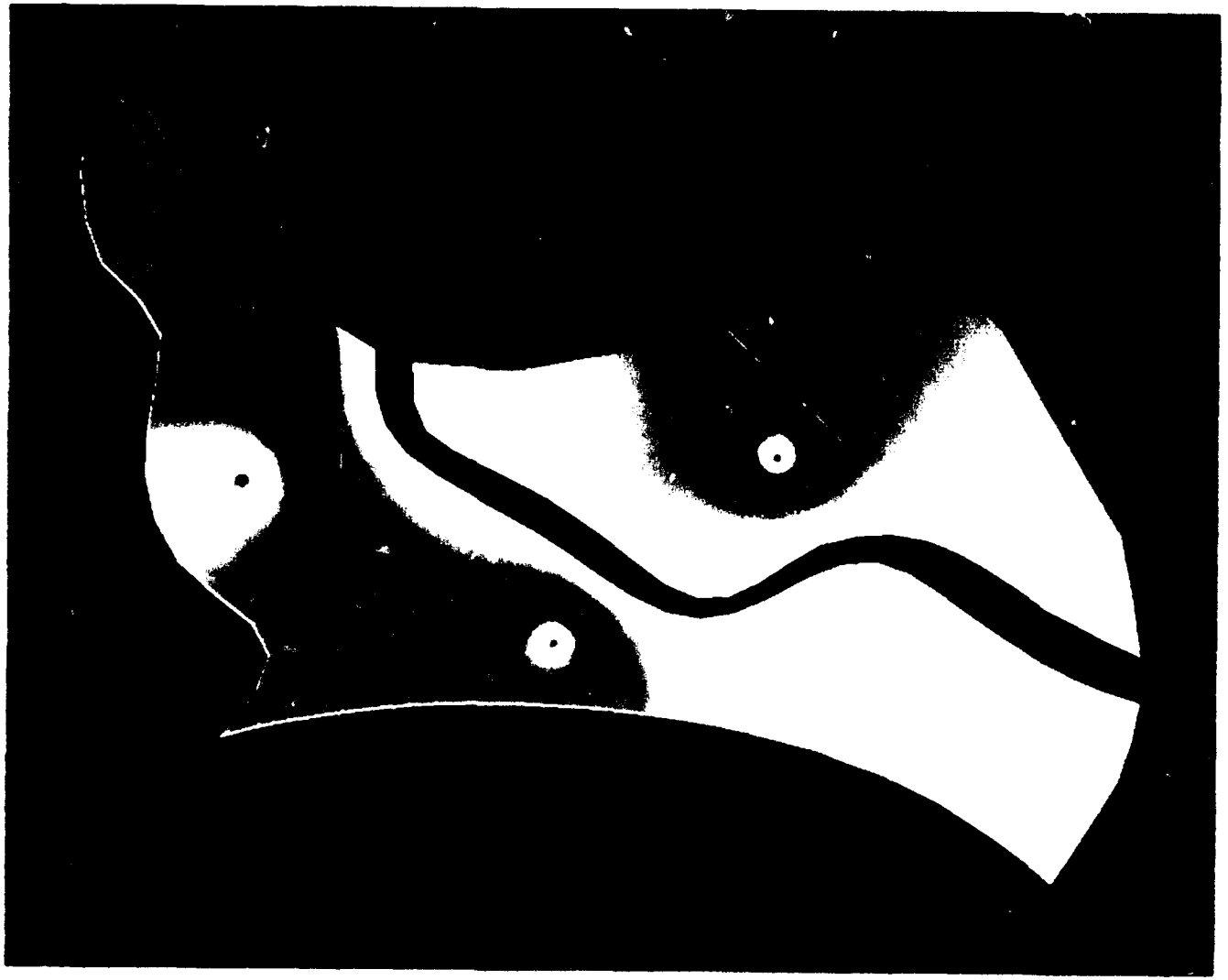


Figure 76. Homogeneous, Isotropic Medium

An important aspect of groundwater flow is the amount of flow from the wells. The three wells pump a total of 512.7 cfm, which is within their range of capacity. Figure 77 shows the color contour plot for unconfined flow. Now, the less pervious region with the two wells has more head loss which is correct. The free surface algorithms can be tested by viewing the free surface at a well. Figure 78 shows a portion of the grid near one of the wells. The results, including the exit line, look as they should. Many other plots can be obtained, but these are left to the interested user of the program to generate.

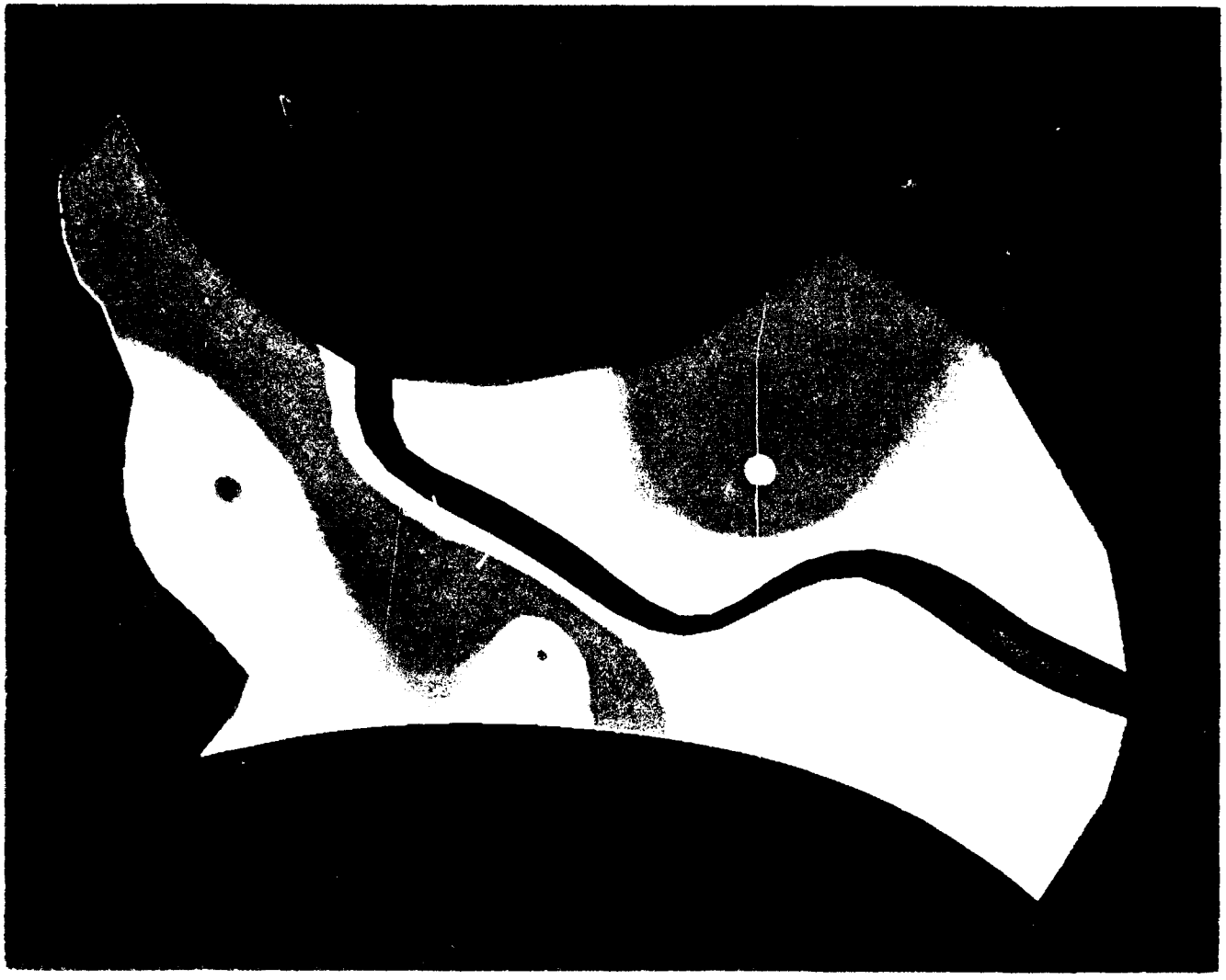


Figure 77. Actual Medium

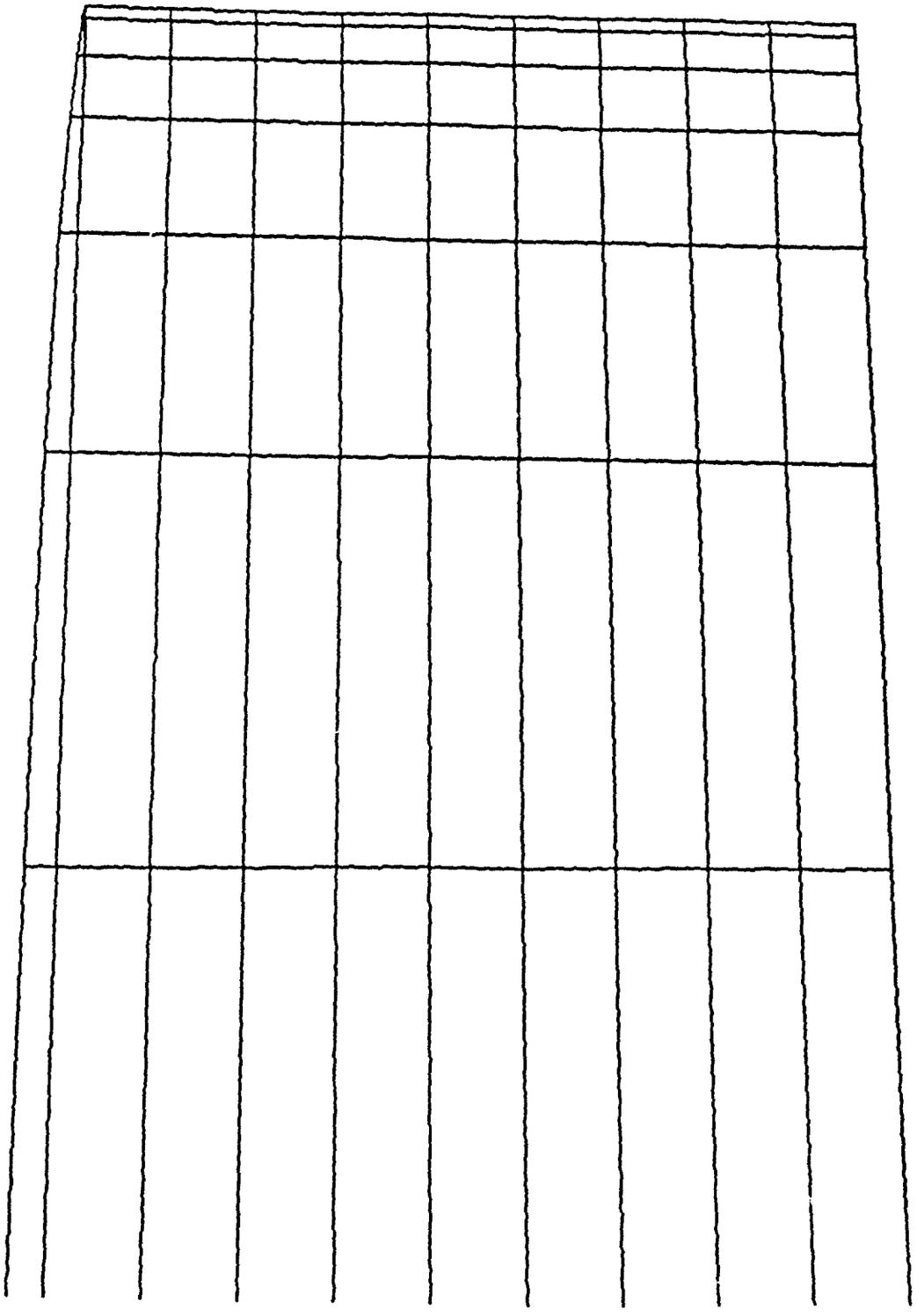


Figure 78. Free Surface at Well

CHAPTER VII

SUMMARY AND CONCLUSION

The techniques and tools ordinarily used by aerospace engineers were successfully applied to 2-D and 3-D seepage and groundwater modeling.

2-D Flow Nets

First, the techniques used to generate an orthogonal grid were broadened in an innovative way to produce computer generated flow nets of superior quality to those from other techniques. When compared with less theoretically based algorithms, the results using the author's work proved to be significantly more accurate.

3-D Modeling

Next, computational techniques using the FEM were developed for 3-D flow applications to apply correct boundary conditions for unconfined flow and dodge the problems that occur at exit lines. The avoidance of a complicated 2-D surface extrapolation for the exit line proved especially helpful. This allowed the successful completion of a 3-D seepage and groundwater model based on the internationally known 2-D seepage package developed by this author.

A very innovative technique and program were next developed to allow the user to first generate pieces of structured grid using EAGLE and then combine the pieces in an efficient manner to produce the unstructured FEM grid. This has several advantages:

1. It makes it easy to apply boundary conditions.
2. Decoupling the geometry makes it easier and faster to generate the resulting FEM grid.
3. Boundaries between blocks only have to match. Cuts and differences in computational coordinates are automatically fixed.
4. The grids are very aesthetic, which is important when showing work to upper management.

Finally, the structure of the original pieces is preserved and output is placed in FAST format for later visualization. The innovative aspect of this work is that for unconfined flow problems the grid collapses in an arbitrary manner to the free surface, and the structure is difficult to maintain. In fact, the only thing that is used is the number of blocks and the dimensions of each block to reconstruct both the collapsed grid and results files.

All the above tools were then applied to the 16-block problem of groundwater flow in an aquifer. It was gratifying to see that the entire process really works!

REFERENCES

Aalto, J. (1984). Finite Element Seepage Flow Nets. International Journal for Numerical and Analytical Methods in Geophysics, 8, 297-303.

Anderson, Dale A., Tannehill, John C., and Pletcher, Richard H. (1984). Computational Fluid Mechanics and Heat Transfer, pp. 182-183. New York: Hemisphere Publishing Corporation.

Bancroft, Gordon, Kelaita, Paul, McCabe, Kevin, Merritt, Fergus, Plessel, Todd, Globus, Al, and Semans, John. (1991). Flow Analysis Software Toolkit (FAST). Sterling Federal Systems, Inc., NASA Ames Research Center, Moffett Field, CA.

Bathe, K. J., and Khashgoftaar, M. R. (1979). Finite Element Free Surface Analysis without Mesh Iteration. International Journal Of Numerical And Analytical Methods in Geomechanics, 3, 13-22.

Biedenharn, Debra G., and Tracy, Fred T. (1987). Finite Element Method Package for Solving Steady-State Seepage Problems. Technical Report ITL-87-6, US Army Engineer Waterways Experiment Station, Vicksburg, MS.

Christian, John T. (1980a). Flow Nets by the Finite Element Method. Ground Water, 18(2).

_____. (1980b). Flow Nets form Finite Element Data. International Journal for Numerical and Analytical Methods in Geomechanics, 4(2).

_____. (19??). Geotechnical Use of Finite Element Flow Analyses. Annual Convention of American Society Civil Engineers.

_____. (1987). Establishing Boundary Condition for Stream Function Contouring of Finite Element Results. Microsoftware for Engineers, 3(2).

Cook, Robert D. (1981). Concepts and Applications of Finite Element Analysis, pp. 113-147. New York: Wiley & Sons, Inc.

Crowder and McCuskey. (1864). Topics in Higher Analysis, pp. 462-467. New York: The MacMillian Company.

Desai, C. S. (1970). Seepage in Mississippi River Banks; Analysis of Transient Seepage using Viscous Flow Model And Numerical Methods. Miscellaneous Paper S-70-3, Report 1, US Army Engineer Waterways Experiment Station, Vicksburg, MS.

DeWiest, R. J. M. (1966). On the Storage Coefficient and the Equations of Groundwater Flow. Journal of Geophysical Research, 71(4), 1117-1122.

Finn, W. D. (1967). Finite-Element Analysis of Seepage through Dams. Journal, Soil Mechanics and Foundations Division, American Society of Civil Engineers, 93(SM6), 44-48.

France, P. W., et al. (1971). Numerical Analysis of Free Surface Seepage Problems. Journal, Irrigation and Drainage Division, American Society of Civil Engineers, 97(IR1), 165-179.

Freeze, Allan R., and Cherry, John A. (1979). Groundwater, p. 172, Englewood Cliffs, NJ: Prentice-Hall, Inc.

Hall, R. L., Tracy, F. T., and Radhakrishnan, N. (1975). Two-Dimensional and Three-Dimensional Seepage Problems using the Finite Element Method. Miscellaneous Paper K-75-6, US Army Engineer Waterways Experiment Station, Vicksburg, MS.

Issacs, L. T., and Mills, K. G. (1972). Numerical Analysis of Free Surface Seepage Problems (Discussion). Journal, Irrigation and Drainage Division, American Society of Civil Engineers, 98(IR1), 150-151.

Maasland, M. (1957). Soil Anisotrophy and Land Drainage. J. N. Luthin (Ed.), Drainage of Agricultural Lands (pp. 218-228). American Society of Agronomy, Madison, WI.

Meyer, C. F., and Kleinecke, D. C. (1968). Development of Capabilities for Mathematical Modeling of Groundwater Flow by Use of Digital Computers. General Electric Co., TEMPO, Santa Barbara, CA.

Muskat, M. (1946). The Flow of Homogeneous Fluids through Porous Media, pp. 263-268. Ann Arbor, MI: J. W. Edwards, Inc.

Neuman, S. P., and Witherspoon, P. A. (1970). Finite Element Method of Analyzing Steady Seepage with a Free Surface. Water Resources Research, 6(3), 1376-1382.

Press, William H., Flannery, Brian P., Teukolsky, Saul A., and Vetterling, William T. (1989). Numerical Recipes, The Art of Scientific Computing, pp. 177. New York: Cambridge University Press.

Taylor, R. L., and Brown, C. R. (1967). Darcy Flow Solutions with a Free Surface. Journal, Hydraulics Division, American Society of Civil Engineers, 93(HY2), 25-33.

Thompson, J. F. (1987). A Composite Grid Generation Code for General 3-D Region. American Institute of Aeronautics and Astronautics 25th Aerospace Science Meeting. Reno, NV.

_____ and Gatlin, B. (1988a). Program EAGLE User's Manual, Volume 1: Introduction and Grid Applications. USAF Armament Laboratory Technical Report AFATL-TR-88-117, Eglin AFB, FL.

_____ and Gatlin, B. (1988b). Program EAGLE User's Manual, Volume 2: Surface Generation Code. USAF Armament Laboratory Technical Report AFATL-TR-88-117, Eglin AFB, FL.

_____ and Gatlin, B. (1988c). Program EAGLE User's Manual, Volume 3: Grid Generation Code. USAF Armament Laboratory Technical Report AFATL-TR-88-117, Eglin AFB, FL.

_____, Warsi, Z. U. A., and Mastin, C. Wayne. (1985). Numerical Grid Generation, Foundation and Applications, pp. 338-352. New York: Elsevier Science Publishing Co., Inc.

Told, David K. (1959). Ground Water Hydrology, pp. 100-103. New York: Wiley & Sons, Inc.

Tracy, Fred T. (1973a). A Three-Dimensional Finite Element Program for Steady-State and Transient Seepage Problems. Miscellaneous Paper K-73-3, US Army Engineer Waterways Experiment Station, Vicksburg, MS.

_____. (1973b). A Plane and Axisymmetric Finite Element Program for Steady-State and Transient Seepage Problems. Miscellaneous Paper K-73-4, US Army Engineer Waterways Experiment Station, Vicksburg, MS.

_____. (1977a). An Interactive Graphics Finite Element Method Grid Generator for Two-Dimensional Problems. Miscellaneous Paper K-77-5, US Army Engineer Waterways Experiment Station, Vicksburg, MS.

_____. (1977b). An Interactive Graphics Post-Processor for Finite Element Method Results. Miscellaneous Paper K-77-4, US Army Engineer Waterways Experiment Station, Vicksburg, MS.

_____. (1977c). Graphical Pre- and Post-Processor for 2-Dimensional Finite Element Method Programs. ACM SIGGRAPH '77 Proceedings, 11(2), 8-17.

_____. (1983). User's Guide for a Plane and Axisymmetric Finite Element Program for Steady-State Seepage Problems. Instruction Report K-83-4, US Army Engineer Waterways Experiment Station, Vicksburg, MS.

_____. (1988). Finite Element Method (FEM) Pre- and Postprocessing on PC's. Corps-wide Structural Engineering Conference Meeting, St. Louis, MO.

Tracy, Fred, T., and Radhakrishnan, N. (1989). Automatic Generation of Seepage Flow Nets by Finite Element Method. Journal of Computing in Civil Engineering, American Society Civil Engineers, 3(3), 268-284.

_____ and Wade, Alberta M. (1980). A Three-Dimensional Finite Element Method Data Edit Program. Miscellaneous Paper K-80-3, US Army Engineer Waterways Experiment Station, Vicksburg, MS.

Wilson, E. L. (1969). Elastic Dynamic Response of Axisymmetric Structures. Report No. 69-2, Structural Engineering Laboratory, College of Engineering, University of California, Berkeley, CA.

Zienkiewicz, O. C. (1971). The Finite Element Method in Engineering Science, 2nd ed. New York: McGraw-Hill.

APPENDIX A

FREE SURFACE SUBROUTINES

This appendix contains the listing and description of three subroutines which are a FORTRAN implementation of the free surface algorithms developed as part of this work.

Subroutine FREES

Subroutine FREES determines the final position of the free surface, and its FORTRAN listing follows.

```
      SUBROUTINE FREES
C
C
C          THIS SUBROUTINE COMPUTES THE POSITION OF THE FREE
C          SURFACE.
C
C
C          DESCRIPTION OF NBC VALUES OF 100 OR GREATER.
C
C          100 - ABOVE THE FS WITH FLOW NOT GIVEN
C          200 - ABOVE THE FS WITH FLOW GIVEN
C          300 - ON FS WITH FLOW NOT GIVEN
C          400 - ON FS WITH FLOW GIVEN
C          500 - BELOW FS WITH FLOW NOT GIVEN
C          600 - BELOW FS WITH FLOW GIVEN
C
C
      COMMON / GRID1 / NUMNP, NUMEL, NUMMAT, X(1000),
     & Y(1000), Z(1000), FX(1000), NBC(1000), FLOW(1000),
     & HLAST(1000)
      COMMON / GRID2 / XK1(12), XK2(12), XK3(12),
     & NP(9, 900), THETA(900), PHI(900), PSI(900)
      COMMON / FREE / FRX(1000), FRY(1000), FRZ(1000),
     & COUNT(1000)
      COMMON  / BANARG / MBAND, NUMBLK, R(1000), C(120,60),
     & ND, ND2
      DIMENSION IADJ(8, 3)
      DATA IADJ /2, 3, 4, 1, 6, 7, 8, 5, 4, 1, 2, 3, 8, 5,
     & 6, 7, 5, 6, 7, 8, 1, 2, 3, 4/
C
C          INITIALIZE DATA.
C
      DO 100 I = 1, NUMNP
      COUNT(I) = 1.E30
      FRX(I) = 0.
      FRY(I) = 0.
      FRZ(I) = 0.
  100 CONTINUE
```

```
C
C          DETERMINE THE FREE SURFACE NODES.
C
      DO 190 N = 1, NUMEL
C
      DO 180 I = 1, 8
C
      N1 = NP(I, N)
      II = IADJ(I, 1)
      N2 = NP(II, N)
      II = IADJ(I, 2)
      N3 = NP(II, N)
      II = IADJ(I, 3)
      N4 = NP(II, N)
      PH1 = R(N1) - Z(N1)
C
      IF (PH1) 120, 110, 180
C
C          FIX ALL BUT NBC = 2 NODE.
C
  110 IF (NBC(N1).EQ.2) GO TO 180
      FRX(N1) = X(N1)
      FRY(N1) = Y(N1)
      FRZ(N1) = Z(N1)
      COUNT(N1) = 0.
      GO TO 180
C
C          CONSIDER LINE SEGMENT N1-N2.
C
  120 CALL FSPT(N1, N2)
C
C          CONSIDER LINE SEGMENT N1-N3.
C
      CALL FSPT(N1, N3)
C
C          CONSIDER LINE SEGMENT N1-N4.
C
      CALL FSPT(N1, N4)
C
  180 CONTINUE
C
  190 CONTINUE
C
C          SET FLAGS AND MOVE FREE SURFACE NODES.
C
      DO 300 I = 1, NUMNP
C
      NBI = NBC(I)
      IF (COUNT(I).LT.1.E29) THEN
         NBI = 300
         IF (NBC(I).NE.0) NBI = 400
         COUNT(I) = 1.
```

```
            X(I) = FRX(I)
            Y(I) = FRY(I)
            Z(I) = FRZ(I)
            R(I) = Z(I)
C
C               FIX FLOW FOR NBC = -1.
C
            IF (NBC(I).EQ.-1) FLOW(I) = FX(I)
         ELSE
            COUNT(I) = 0.
            IF (R(I).LT.Z(I)) THEN
               NBI = 100
               IF (NBC(I).NE.0) NBI = 200
            ELSE
               NBI = 500
               IF (NBC(I).NE.0) NBI = 600
            ENDIF
         ENDIF
C
         NBC(I) = NBI
C
  300 CONTINUE
C
C            FIX ELEMENTS CLOSE TO THE FREE SURFACE.
C
         CALL FIXEL
C
         RETURN
         END
```

The codes 100-600 in the comments are flags used to determine how the node is to be considered upon print-out of results. A node is below, on, or above the free surface with flow given or not given. Variables FRX, FRY, and FRZ contain the coordinates of the new position of a free surface node, and the value of COUNT determines if the node is a free surface node. COUNT is initialized to 10^{30}, and if it is set to a smaller value, the node is a free surface node. Do loops 180 and 190 check each line segment of each element to find a line segment where the pressure head is less than zero on the first node and greater than zero on

the second node. When such a line segment is found, subroutine FSPT is called to compute the new position of the free surface node.

The next section (do loop 300) does three things:

1. Changes the value of COUNT to be 1 for a free surface node and 0 otherwise.

2. Puts in the array NBC the print codes (100-600) describing the status of each node.

3. Changes the (x, y, z) coordinates of the free surface nodes to their respective new free surface values.

Finally, subroutine FIXEL is called to fix the elements to be either all below or all above the free surface.

Subroutine FSPT

Subroutine FSPT looks for a free surface point between nodes N1 and N2. Its listing is now given.

```
      SUBROUTINE FSPT(N1, N2)
C
C
C         THIS SUBROUTINE CHECKS LINE SEGMENT M1-N2 FOR A
C         POTENTIAL FREE SURFACE NODE.  IF SO, THE NEW
C         POSITION OF NODE N1 IS COMPUTED.
C
C
      COMMON / GRID1 / NUMNP, NUMEL, NUMMAT, X(1000),
     & Y(1000), Z(1000), FX(1000), NBC(1000), FLOW(1000),
     & HLAST(1000)
      COMMON / GRID2 / XK1(12), XK2(12), XK3(12),
     & NP(9, 900), THETA(900), PHI(900), PSI(900)
      COMMON / FREE / FRX(1000), FRY(1000), FRZ(1000),
     & COUNT(1000)
      COMMON  / BANARG / MBAND, NUMBLK, R(1000), C(120,60),
     & ND, ND2
C
C         SET EXIT POINT PARAMETER.
C
      BETA = .8
C
```

```
      PH1 = R(N1) - Z(N1)
      PH2 = R(N2) - Z(N2)
C
      IF (PH2) 140, 100, 110
C
C         COMPUTE S USING THE SPECIAL EXIT LINE COMPUTATION.
C
  100 IF ((NBC(N1).NE.2).OR.(NBC(N2).NE.2)) GO TO 140
      DZZ = Z(N2) - Z(N1)
      IF (DZZ.EQ.0.) THEN
         S = 1.
      ELSE
         S0 = PH1 / DZZ
         S = (1 - BETA) * S0 + BETA
      ENDIF
      GO TO 120
C
C         DON'T LET A SURFACE OF SEEPAGE NODE BE MOVED OFF
C         THE SURFACE OF SEEPAGE.
C
  110 IF (NBC(N1).EQ.2) GO TO 140
C
C         COMPUTE S USING THE FREE SURFACE NODE COMPUTATION.
C
      S = PH1 / (PH1 - PH2)
C
  120 IF ((S.LT.0.).OR.(S.GT.1.)) S = 1.
      DX = (X(N2) - X(N1)) * S
      DY = (Y(N2) - Y(N1)) * S
      DZ = (Z(N2) - Z(N1)) * S
      RR = DX * DX + DY * DY + DZ * DZ
C
C         KEEP THE ONE WITH THE SMALLEST MOVEMENT.
C
      IF (RR.GE.COUNT(N1)) GO TO 140
      FRX(N1) = X(N1) + DX
      FRY(N1) = Y(N1) + DY
      FRZ(N1) = Z(N1) + DZ
      COUNT(N1) = RR
C
  140 RETURN
      END
```

The basic idea of this subroutine is to compute the value of the parameter s, where

$$0 \leq s \leq 1$$

such that the pressure head is zero. That is, use Equation 4.28 for a node not on the exit face or Equations 4.29 and 4.30 for a node on the exit face (NBC = 2). The changes in coordinate values (variables DX, DY, and DZ) required to move point N1 to the new free surface position are then computed. Distance squared (in variable RR) is then computed from these values and compared with the current value of COUNT. If RR is less than COUNT, the position of the free surface is computed along this line segment and stored in FRX, FRY, and FRZ. COUNT is then updated to RR. This position of the free surface is kept for node N1 until a smaller RR comes along as a result of considering other line segments.

Subroutine FIXEL

This subroutine performs the iterative process of redefining the coordinates of the node points so elements are either all below the free surface or collapsed onto it. Its listing is given.

```
      SUBROUTINE FIXEL
C
C
C         THIS SUBROUTINE FIXES THE ELEMENTS CROSSED BY THE
C         FREE SURFACE SO AN ELEMENT IS EITHER ALL BELOW THE
C         FREE SURFACE OR COLLAPSED ONTO IT.
C
C
      COMMON / GRID1 / NUMNP, NUMEL, NUMMAT, X(1000),
     & Y(1000), Z(1000), FX(1000), NBC(1000), FLOW(1000),
     & HLAST(1000)
      COMMON / GRID2 / XK1(12), XK2(12), XK3(12), NP(9,
     & 900), THETA(900), PHI(900), PSI(900)
```

```
      COMMON / FREE / FRX(1000), FRY(1000), FRZ(1000),
     & COUNT(1000)

      COMMON  / BANARG / MBAND, NUMBLK, R(1000), C(120,60),
     & ND, ND2
      DIMENSION ISEG(2, 12)
      DATA ISEG /1, 2, 2, 3, 3, 4, 4, 1, 1, 5, 2, 6, 3, 7,
     & 4, 8, 5, 6, 6, 7, 7, 8, 8, 5/
C
C         KEEP ITERATING UNTIL THERE IS NO CHANGE.
C
      ICOUNT = 0
C
  100 ICHANG = 0
      ICOUNT = ICOUNT + 1
      IF (ICOUNT.GT.200) THEN
         PRINT*, ' AFTER 200 ITERATIONS THE SUBROUTINE',
     &           ' FIXEL ALGORITHM HAD NOT CONVERGED.'
         RETURN
      ENDIF
C
      DO 600 N = 1, NUMEL
C
C         CHECK IF THE FREE SURFACE INTERSECTS THE ELEMENT.
C
      PMIN = 1.E30
      PMAX = - PMIN
      DO 200 I = 1, 8
      II = NP(I, N)
      P = R(II) - Z(II)
      PMIN = AMIN1(PMIN, P)
      PMAX = AMAX1(PMAX, P)
  200 CONTINUE
C
C         IF THE ELEMENT IS ABOVE THE FREE SURFACE AND DOES
C         NOT TOUCH IT, DON'T BOTHER WITH IT.
C
      IF (PMAX.LT.0.) GO TO 600
C
C         IF THE ELEMENT IS BELOW THE FREE SURFACE, DON'T
C         BOTHER WITH IT.
C
      IF (PMIN.GE.0.) GO TO 600
C
C         THE ELEMENT IS CROSSED BY THE FREE SURFACE, SO
C         CHECK EACH LINE SEGMENT.
C
      DO 500 J = 1, 12
C
      II = ISEG(1, J)
      N1 = NP(II, N)
      II = ISEG(2, J)
```

```
      N2 = NP(II, N)
C
C          IF LINE SEGMENT N1-N2 IS A FREE SURFACE NODE AND A
C          NODE ABOVE THE FREE SURFACE, COLLAPSE THE NODE
C          ABOVE THE FREE SURFACE TO THE EXIT POINT NODE.
C
      DO 300 K = 1, 2
C
C          CONSIDER BOTH N1-N2 AND N2-N1.
C
      IF (K.EQ.2) THEN
         NN = N1
         N1 = N2
         N2 = NN
      ENDIF
C
      IF ((COUNT(N1).GT.0.).AND.(R(N2).LT.Z(N2))) THEN
         X(N2) = X(N1)
         Y(N2) = Y(N1)
         Z(N2) = Z(N1)
         R(N2) = R(N1)
         COUNT(N2) = 2.
C
C          FLAG THAT A CHANGE HAS BEEN MADE.
C
         ICHANG = 1
      ENDIF
C
  300 CONTINUE
C
  500 CONTINUE
C
  600 CONTINUE
C
C          IF A CHANGE WAS MADE, REPEAT THE PROCESS.
C
      IF (ICHANG.EQ.1) GO TO 100
C
      RETURN
      END
```

First, note that a maximum of 200 iterations are allowed. Next, the maximum and minimum pressure heads for the element are computed. If they are all greater than or equal to zero, the element is completely under the free surface. If they are all less than zero, the element is

completely above the free surface and does not touch it. In each of these cases the element needs no further action. Otherwise, the algorithm given at the end of Chapter IV is executed.

Here, each line segment of the element is considered for the first node being a free surface node and the second node of the line segment being above the free surface (pressure head less than zero). When such a line segment is found, node 2 is collapsed to node 1 of the line segment, and node 2 is flagged as a special free surface node (COUNT = 2).

After going through all the elements (do loop 600) with no further collapses occurring, the iteration process is terminated.

APPENDIX B

CONVERSION TO FEM FORMAT

This appendix documents the program written to combine blocks generated by the EAGLE program into a consolidated FEM grid. This approach makes it very easy to generate separate pieces and then combine them later. This approach also makes it significantly easier than usual to apply boundary conditions.

MAIN Program

The MAIN program is the driver for the rest of the program and is given:

```
C          THIS PROGRAM COMBINES EAGLE BLOCKS INTO ONE BIG
C          FEM GRID.
C
C
       PARAMETER (ND = 1000)
       COMMON /OUTPUT/ X(ND), Y(ND), Z(ND), IBC(ND), BV(ND),
      & IX(9, ND), THETA(ND), PHI(ND), PSI(ND), NUMNP, NUMEL,
      & NUMMAT, NODVEL
       CHARACTER COM * 3, ANAME * 20
C
       NUMNP = 0
       NUMEL = 0
       NUMMAT = 0
       NODVEL = 0
C
       OPEN (7, ACCESS='DIRECT', RECL=16)
       OPEN (8, FORM='UNFORMATTED')
       OPEN (9, FORM='UNFORMATTED')
C
C          PROCESS THE COMMANDS
C
  100  PRINT*, ' '
       PRINT*, 'COMMAND?'
       READ (*, 110) COM
  110  FORMAT(A)
       IF ((COM.EQ.'INP').OR.(COM.EQ.'inp')) GO TO 400
       IF ((COM.EQ.'COM').OR.(COM.EQ.'com')) GO TO 200
       IF ((COM.EQ.'OUT').OR.(COM.EQ.'out')) GO TO 500
       IF ((COM.EQ.'BAN').OR.(COM.EQ.'ban')) GO TO 300
       IF ((COM.EQ.'BC ').OR.(COM.EQ.'bc ')) GO TO 600
       IF ((COM.EQ.'CLE').OR.(COM.EQ.'cle')) GO TO 700
       IF ((COM.EQ.'END').OR.(COM.EQ.'end')) GO TO 800
```

```
      PRINT 120
  120 FORMAT(/ ' INP - INPUT A SINGLE FILE.' /
     &         ' COM - COMBINE SEVERAL FILES.' /
     &         ' OUT - OUTPUT A FILE.' /
     &         ' BAN - BANDWIDTH MINIMIZE.' /
     &         ' BC  - APPLY BOUNDARY CONDITIONS.' /
     &         ' CLE - CLEAR PREVIOUS WORK.' /
     &         ' END - END EXECUTION OF PROGRAM.')
      GO TO 100
C
C         COMBINE THE GRIDS.
C
  200 PRINT*, ' '
      PRINT*, 'NUMBER OF FILES'
      READ (*, *) NOFILE
      CALL COMBIN(NOFILE, IBL)
      GO TO 100
C
C         BANDWIDTH MINIMIZE THE GRID.
C
  300 CALL ADJAC
      CALL BANMIN
      GO TO 100
C
C         READ A GRID.
C
  400 CALL COMBIN(1, IBL)
      GO TO 100
C
C         OUTPUT THE GRID.
C
  500 PRINT*, ' '
      PRINT*, 'OUTPUT FILE NAME FOR SEEPAGE/GROUNDWATER',
     &        ' MODEL (WITHOUT EXTENSIONS)?'
      READ (*, 110) ANAME
      NN = NONBLK(ANAME)
      OPEN (2, FILE=ANAME(1 : NN) // '.sep',
     &      STATUS='UNKNOWN')
      OPEN (3, FILE=ANAME(1 : NN) // '.dim',
     &      STATUS='UNKNOWN')
      CALL OUTFEM(IBL)
      CLOSE (2)
      CLOSE (3)
      GO TO 100
C
C         APPLY BOUNDARY CONDITIONS.
C
  600 PRINT*, ' '
      PRINT*, 'BLOCK NUMBER, I1B, J1B, K1B, I2B, J2B, K2B,',
     &        ' IBC, BV?'
      READ (*, *) IBLNO, I1B, J1B, K1B, I2B, J2B, K2B, NBC,
     & BCVAL
```

```
      CALL BC(IBLNO, I1B, J1B, K1B, I2B, J2B, K2B, NBC,
     & BCVAL)
      GO TO 100
C
C          CLEAR PREVIOUS WORK.
C
  700 NUMNP = 0
      NUMEL = 0
      NUMMAT = 0
      NODVEL = 0
      REWIND 7
      REWIND 8
      REWIND 9
      GO TO 100
C
C          TERMINATE PROCESSING.
C
  800 CALL DONE
      STOP
      END
```

There are seven commands provided to accomplish the task of creating a single FEM grid from individual blocks as follows:

1. Input - Input a single EAGLE grid file.
2. Combine - Combine a group of EAGLE grid files.
3. Output - Output the FEM grid.
4. Bandwidth - Minimize the bandwidth of the FEM grid.
5. BC - Apply boundary conditions.
6. Clear - Clear previous work.
7. End - End execution of program.

A description of the subroutines supporting these commands is now given.

Subroutine ADJAC

This subroutine produces an adjacency table giving the nodes that are adjacent to a given node. It is used in the bandwidth minimization process and is as follows:

```
      SUBROUTINE ADJAC
C
C
C         THIS SUBROUTINE COMPUTES AN ADJACENCY TABLE FOR
C         THE GENERATED MESH.
C
C
      PARAMETER (ND = 1000)
      COMMON /OUTPUT/ X(ND), Y(ND), Z(ND), IBC(ND), BV(ND),
     & IX(9, ND), THETA(ND), PHI(ND), PSI(ND), NUMNP, NUMEL,
     & NUMMAT, NODVEL
      COMMON /ADJ/ JZ(ND, 9)
      DIMENSION IADJ(8, 3)
      DATA IADJ /2, 3, 4, 1, 6, 7, 8, 5, 4, 1, 2, 3, 8, 5,
     & 6, 7, 5, 6, 7, 8, 1, 2, 3, 4/
C
C
C         JZ  -  ARRAY CONTAINING THE NEIGHBORING NODES FOR
C                EACH NODE (ADJACENCY TABLE).  NO MORE THAN
C                EIGHT NEAREST NEIGHBORS MAY EXIST FOR A
C                GIVEN NODE.
C
C
C         ZERO THE JZ ARRAY.
C
      DO 200 N = 1, NUMNP
      JZ(N, 9) = 0
  200 CONTINUE
C
C         CONSIDER EACH ELEMENT.
C
      DO 1300 N = 1, NUMEL
C
C         CHECK EACH NODE OF EACH ELEMENT.
C
      DO 1200 L = 1, 8
C
      NODP = IX(L, N)
C
      DO 1100 M = 1, 3
C
      LC = IADJ(L, M)
      NODS = IX(LC, N)
      KK = JZ(NODP, 9)
C
C         PLACE THE SECONDARY NODE INTO THE ROW OF JZ
C         ORRESPONDING TO THE PRIMARY NODE, IF IT HASN'T
C         ALREADY BEEN DONE.
C
      IF (KK) 700, 700, 900
C
  700 JZ(NODP, 1) = NODS
```

```
      JZ(NODP, 9) = 1
      GO TO 1100
C
  900 DO 1000 K = 1, KK
      IF (JZ(NODP, K).EQ.NODS) GO TO 1100
 1000 CONTINUE
      K = KK + 1
      IF (K.GT.8) THEN
         PRINT*, 'THE NODE', NODP, ' HAS MORE THAN EIGHT',
     &           ' LINES GOING INTO IT.'
         CALL DONE
         STOP
      ENDIF
 1050 JZ(NODP, K) = NODS
      JZ(NODP, 9) = K
C
 1100 CONTINUE
C
 1200 CONTINUE
C
 1300 CONTINUE
C
      RETURN
      END
```

Subroutine BANMIN

This subroutine does the bandwidth minimization for the FEM grid. Its listing is now given:

```
      SUBROUTINE BANMIN
C
C
C         THIS SUBROUTINE REDUCES THE BANDWIDTH OF A FEM
C         CONFIGURATION.
C
C
      PARAMETER (ND = 1000)
      COMMON /OUTPUT/ X(ND), Y(ND), Z(ND), IBC(ND), BV(ND),
     & IX(9, ND), THETA(ND), PHI(ND), PSI(ND), NUMNP, NUMEL,
     & NUMMAT, NODVEL
      COMMON /ADJ/ JZ(ND, 9)
      COMMON /SCRAT/ NODE(ND), NNS(ND), NOS(ND)
C
C
C         NOS  -  ARRAY CONTAINING THE NEW NODE NUMBERS WITH
C                 THE OLD NODE NUMBERS AS SUBSCRIPTS.
C
```

```
C          NNS  -  ARRAY CONTAINING THE OLD NODE NUMBERS WITH
C                 THE NEW NODE NUMBERS AS SUBSCRIPTS.
C
C          NODE  -  ARRAY CONTAINING THE BEST NOS ARRAY.
C
C          NBWMIN  -  MINIMUM MAXIMUM DIFFERENCE FOUND.
C
C
C          COMPUTE THE MAXIMUM DIFFERENCE OF THE CURRENT
C          CONFIGURATION.
C
      NBW = 0
      DO 200 N = 1, NUMEL
      DO 100 J = 1, 7
      J1 = IX(J, N)
      KK = J + 1
      DO 60 K = KK, 8
      K1 = IX(K, N)
      KD = IABS(J1 - K1)
      NBW = MAX0(NBW, KD)
   60 CONTINUE
  100 CONTINUE
  200 CONTINUE
C
      PRINT 205, NBW
  205 FORMAT (/ 'INITIAL MAXIMUM DIFFERENCE = ', I5)
C
C          DETERMINE THE SMALLEST MAXIMUM DIFFERENCE
C          POSSIBLE.
C
      NBWMIN = NBW
C
C          ITERATE TO A SOLUTION.
C
      NOITER = MINO(NUMNP, 20)
      DO 220 N = 1, NUMNP
      NODE(N) = N
  220 CONTINUE
C
      DO 1100 L = 1, NOITER
C
C          INITIALIZE.
C
      DO 300 N = 1, NUMNP
      NNS(N) = 99999
      NOS(N) = 0
  300 CONTINUE
C
      K = 1
      NNS(1) = L
      NOS(L) = 1
      NDIFF = 0
```

```
C
C          PROCESS THE ADJACENCY TABLE.
C
      DO 800 N = 1, NUMNP
C
      ISUB = NNS(N)
      NUM = JZ(ISUB, 9)
C
      DO 700 M = 1, NUM
C
      NBR = JZ(ISUB, M)
      IF (NOS(NBR)) 400, 400, 700
  400 K = K + 1
      NOS(NBR) = K
      NNS(K) = NBR
C
C          CHECK IF MINIMUM MAXIMUM DIFFERENCE HAS BEEN
C          EXCEEDED.
C
      KD = IABS(N - K)
      NDIFF = MAX0(NDIFF, KD)
      IF (KD - NBWMIN) 500, 500, 1100
C
C          CHECK IF ITERATION IS FINISHED.
C
  500 IF (K - NUMNP) 700, 900, 900
C
  700 CONTINUE
C
  800 CONTINUE
C
C          DETERMINE THE MAXIMUM DIFFERENCE OF THE CURRENT
C          CONFIGURATION.
C
  900 NBW = NDIFF
      DO 920 N = 1, NUMEL
      DO 910 J = 1, 7
      J1 = IX(J, N)
      J1 = NOS(J1)
      KK = J + 1
      DO 905 K = KK, 8
      K1 = IX(K, N)
      K1 = NOS(K1)
      KD = IABS(J1 - K1)
      NBW = MAX0(NBW, KD)
  905 CONTINUE
  910 CONTINUE
  920 CONTINUE
C
C          CHECK FOR THE BEST CONFIGURATION.
C
      IF (NBWMIN - NBW) 1100, 1100, 950
```

```
C
  950 NBWMIN = NBW
      DO 1000 N = 1, NUMNP
      NODE(N) = NOS(N)
 1000 CONTINUE
C
      PRINT 1010, L, NBWMIN
 1010 FORMAT ('ITERATION = ', I5, 10X, 'BEST MAXIMUM'
     & ' DIFFERENCE = ', I5)
C
 1100 CONTINUE
C
C          SHUFFLE THE DATA.
C
      DO 1200 N = 1, NUMNP
      N1 = NODE(N)
      NNS(N1) = N
 1200 CONTINUE
C
C          FIX THE NODE DATA.
C
      REWIND 8
      DO 1400 N = 1, NUMNP
      N1 = NNS(N)
      WRITE (8) X(N1), Y(N1), Z(N1), IBC(N1), BV(N1)
 1400 CONTINUE
      REWIND 8
      DO 1500 N = 1, NUMNP
      READ (8) X(N), Y(N), Z(N), IBC(N), BV(N)
 1500 CONTINUE
C
C          FIX THE ELEMENT DATA.
C
      DO 1700 N = 1, NUMEL
      DO 1600 I = 1, 8
      I1 = IX(I, N)
      I2 = NODE(I1)
      IX(I, N) = I2
 1600 CONTINUE
 1700 CONTINUE
C
      RETURN
      END
```

The most difficult aspect of this subroutine is keeping up with the new node numbers (the two arrays NOS and NNS accommodate this). Finally, any number of tries can be made as the starting point to renumber the grid up to the number

of node points. This subroutine currently has the number of iterations as 20.

Subroutine BC

Subroutine BC applies boundary conditions, and its listing is now given:

```
      SUBROUTINE BC(IBLNO, I1B, J1B, K1B, I2B, J2B, K2B,
     & NBC, BCVAL)
C
C
C          THIS SUBROUTINE APPLIES BOUNDARY CONDITIONS TO THE
C          FACES OF BLOCKS.
C
C
      PARAMETER (ND = 1000)
      COMMON /OUTPUT/ X(ND), Y(ND), Z(ND), IBC(ND), BV(ND),
     & IX(9, ND), THETA(ND), PHI(ND), PSI(ND), NUMNP, NUMEL,
     & NUMMAT, NODVEL
      DIMENSION IPICK(2, 2, 2), IFACE(4, 6), JNODE(4)
      DATA IPICK /1, 2, 4, 3, 5, 6, 8, 7/
      DATA IFACE /1, 5, 8, 4, 2, 3, 7, 6, 1, 2, 6, 5, 3, 4,
     & 8, 7, 4, 3, 2, 1, 5, 6, 7, 8/
C
C          GET SIZE OF BLOCK AND NUMBER OF ELEMENTS JUST
C          BEFORE IT BEGINS.
C
      READ (7, REC=IBLNO) I2, J2, K2, NEL
      IROW = I2 - 1
      IPLANE = (J2 - 1) * IROW
C
C          APPLY VARIOUS BOUNDARY CONDITIONS, EXCEPT NBC =
C          -2.
C
      IF (NBC.EQ.-2) GO TO 3050
C
      DO 3000 K = K1B, K2B
C
      IF (K.LT.K2) THEN
         KK = 1
      ELSE
         KK = 2
      ENDIF
C
      DO 2900 J = J1B, J2B
C
      IF (J.LT.J2) THEN
```

```
         JJ = 1
      ELSE
         JJ = 2
      ENDIF
C
      DO 2800 I = I1B, I2B
C
      IF (I.LT.I2) THEN
         II = 1
      ELSE
         II = 2
      ENDIF
C
C          GET NODE NUMBER.
C
      MM = (K - KK) * IPLANE + (J - JJ) * IROW + I - II + 1
     & + NEL
      IPOS = IPICK(II, JJ, KK)
      NN = IX(IPOS, MM)
C
C          DO VARIOUS BOUNDARY CONDITIONS.
C
      IF ((NBC.EQ.1).OR.(NBC.EQ.-1)) THEN
         IBC(NN) = NBC
         BV(NN) = BCVAL
      ELSE IF ((NBC.EQ.2).OR.(NBC.EQ.12)) THEN
         IBC(NN) = NBC
         BV(NN) = Z(NN)
      ENDIF
C
 2800 CONTINUE
C
 2900 CONTINUE
C
 3000 CONTINUE
C
      GO TO 5000
C
C          DO SPECIFIED DISCHARGE VELOCITY FOR A FACE.
C
 3050 IF ((I1B.EQ.1).AND.(I2B.EQ.1)) THEN
         JF = 1
         L1 = J1B
         L2 = J2B - 1
         M1 = K1B
         M2 = K2B - 1
         LFACT = I2 - 1
         MFACT = LFACT * (J2 - 1)
         MST = NEL + 1
      ELSE IF ((I1B.EQ.I2).AND.(I2B.EQ.I2)) THEN
         JF = 2
         L1 = J1B
```

```
            L2 = J2B - 1
            M1 = K1B
            M2 = K2B - 1
            LFACT = I2 - 1
            MFACT = LFACT * (J2 - 1)
            MST = NEL + I2 - 1
         ELSE IF ((J1B.EQ.1).AND.(J2B.EQ.1)) THEN
            JF = 3
            L1 = I1B
            L2 = I2B - 1
            M1 = K1B
            M2 = K2B - 1
            LFACT = 1
            MFACT = (I2 - 1) * (J2 - 1)
            MST = NEL + 1
         ELSE IF ((J1B.EQ.J2).AND.(J2B.EQ.J2)) THEN
            JF = 4
            L1 = I1B
            L2 = I2B - 1
            M1 = K1B
            M2 = K2B - 1
            LFACT = 1
            MFACT = (I2 - 1) * (J2 - 1)
            MST = NEL + (I2 - 1) * (J2 - 2) + 1
         ELSE IF ((K1B.EQ.1).AND.(K2B.EQ.1)) THEN
            JF = 5
            L1 = I1B
            L2 = I2B - 1
            M1 = J1B
            M2 = J2B - 1
            LFACT = 1
            MFACT = I2 - 1
            MST = NEL + 1
         ELSE IF ((K1B.EQ.K2).AND.(K2B.EQ.K2)) THEN
            JF = 6
            L1 = I1B
            L2 = I2B - 1
            M1 = J1B
            M2 = J2B - 1
            LFACT = 1
            MFACT = I2 - 1
            MST = NEL + (I2 - 1) * (J2 - 1) * (K2 - 2) + 1
         ENDIF
C
         DO 3500 M = M1, M2
C
         DO 3400 L = L1, L2
C
C           GET NODE NUMBERS.
C
         MM = (L - 1) * LFACT + (M - 1) * MFACT + MST
         DO 3100 I = 1, 4
```

```
      JPOS = IFACE(I, JF)
      JJ = IX(JPOS, MM)
      IBC(JJ) = -1
      JNODE(I) = JJ
 3100 CONTINUE
C
C          SAVE RESULTS.
C
      NODVEL = NODVEL + 1
      WRITE (9) JNODE, BCVAL
C
 3400 CONTINUE
C
 3500 CONTINUE
C
 5000 RETURN
      END
```

This subroutines is roughly divided into two parts. The first part applies boundary conditions 1, 2, 12, or -1 to a group on nodes, typically a sheet on the surface of the block. The last section determines sets of four nodes required for discharge velocity data and writes them to a temporary file. Since the elements are not shuffled in bandwidth minimization, the proper nodes can be obtained by looking up the correct node of the correct element in the connectivity array. This process is greatly simplified by the arrays IPICK and IFACE.

Subroutine COMBIN

Subroutine COMBIN reads in any number of EAGLE grid files and combines them into a single FEM grid. Its listing follows.

```
      SUBROUTINE COMBIN(NOFILE, IBL)
C
C
C          THIS SUBROUTINE TAKES TWO OR MORE FILES CONTAINING
C          EAGLE GRIDS AND COMBINES THEM INTO ONE GRID.
C
C
      PARAMETER (ND = 1000)
      COMMON /OUTPUT/ X(ND), Y(ND), Z(ND), IBC(ND), BV(ND),
     & IX(9, ND), THETA(ND), PHI(ND), PSI(ND), NUMNP, NUMEL,
     & NUMMAT, NODVEL
      COMMON /SCRAT/ NOD(ND), IREPL(ND), BB(ND)
      CHARACTER ANAME * 20
      DIMENSION I2S(20), J2S(20), K2S(20)
C
      IBL = 0
      IZERO = 0
C
C          LOOP OVER THE NUMBER OF FILES.
C
      DO 1000 M = 1, NOFILE
C
C          READ EAGLE GRID FILE.
C
   10 PRINT 15, M
   15 FORMAT(/ ' FILE NAME NO.', I3, ' ?')
   20 READ (*, 25) ANAME
   25 FORMAT(A)
      OPEN (2, FILE=ANAME, STATUS='OLD', IOSTAT=ISTAT)
      IF (ISTAT.NE.0) THEN
         PRINT*, ' '
         NN = NONBLK(ANAME)
         PRINT*, 'INCORRECT FILE NAME - ', ANAME(1 : NN),
     &           ' - TRY AGAIN.'
         GO TO 20
      ENDIF
C
C          GET THE NUMBER OF BLOCKS.
C
      READ (2, *) NUMBLK
      IF (NUMBLK.GT.20) THEN
         PRINT*, 'NO MORE THAN 20 BLOCKS MAY BE IN',
     &           ' ONE FILE.'
         STOP
      ENDIF
C
C          GET THE GRID DIMENSIONS FOR ALL BLOCKS.
C
      DO 30 L = 1, NUMBLK
      READ (2, *) I2S(L), J2S(L), K2S(L)
   30 CONTINUE
C
```

```
C          LOOP OVER THE NUMBER OF BLOCKS.
C
      DO 900 L = 1, NUMBLK
C
C          GET THE SIZE OF THE BLOCK.
C
      I2 = I2S(L)
      J2 = J2S(L)
      K2 = K2S(L)
      IJ2 = I2 * J2
C
C          OUTPUT BLOCK DATA TO TEMPORARY FILE.
C
      IBL = IBL + 1
      WRITE (7, REC=IBL) I2, J2, K2, NUMEL
C
C          READ THE (X, Y, Z) DATA POINTS.
C
      NUM = I2 * J2 * K2
      ND1 = NUMNP + 1
      NDTOT = NUMNP + NUM
      READ (2, *) (X(I), I = ND1, NDTOT), (Y(I), I = ND1,
     & NDTOT), (Z(I), I = ND1, NDTOT)
C
C          COMPUTE ELEMENT DATA.
C
      NN = NUMNP
      MM = NUMEL
C
C          GET MATERIAL TYPE NUMBER, THETA, PHI, AND PSI.
C
      PRINT 42, L
   42 FORMAT(/ ' FOR BLOCK', I3, ' MATERIAL TYPE NUMBER,
     & THETA, PHI,', ' AND PSI?')
      READ*, MAT, TH, PH, PS
      NUMMAT = MAX0(NUMMAT, MAT)
C
      DO 70 K = 1, K2 - 1
C
      DO 60 J = 1, J2 - 1
C
      DO 50 I = 1, I2 - 1
C
      NN = NN + 1
      MM = MM + 1
C
      IX(1, MM) = NN
      IX(2, MM) = IX(1, MM) + 1
      IX(3, MM) = IX(2, MM) + I2
      IX(4, MM) = IX(3, MM) - 1
C
      DO 45 II = 1, 4
```

```
      IX(II+4, MM) = IX(II, MM) + IJ2
   45 CONTINUE
C
      IX(9, MM) = MAT
      THETA(MM) = TH
      PHI(MM) = PH
      PSI(MM) = PS
C
   50 CONTINUE
C
      NN = NN + 1
C
   60 CONTINUE
C
      NN = NN + I2
C
   70 CONTINUE
C
      NUMEL = MM
C
C         INITIATE THE NODE ARRAY.
C
      DO 220 I = 1, NDTOT
      NOD(I) = I
      IREPL(I) = 0
  220 CONTINUE
C
C         REMOVE DUPLICATE NODES.
C
      IF (L.EQ.1) THEN
         N1 = 1
      ELSE
         N1 = ND1
      ENDIF
C
      DO 300 N = N1, NDTOT
C
      XCK = X(N)
      YCK = Y(N)
      ZCK = Z(N)
      IF (MOD(N, 1000).EQ.0) PRINT*, 'REMOVING DUPLICATE ',
     & 'NODES, N =', N, '   NUMNP =', NUMNP
C
      DO 240 I = 1, N-1
      IF (IREPL(I).EQ.1) GO TO 240
      CK = ABS(X(I) - XCK) + ABS(Y(I) - YCK) + ABS(Z(I) -
     & ZCK)
      IF (CK - 1.E-2) 250, 250, 240
  240 CONTINUE
      GO TO 300
C
  250 IREPL(N) = 1
```

```
      NOD(N) = NOD(I)
      IF (N.EQ.NDTOT) GO TO 300
C
  260 NP1 = N + 1
      DO 270 I = NP1, NDTOT
      NOD(I) = NOD(I) - 1
  270 CONTINUE
C
  300 CONTINUE
C
C         UPDATE THE NODE DATA.
C
      LL = NUMNP
      DO 320 I = ND1, NDTOT
      IF (IREPL(I).EQ.1) GO TO 320
      LL = LL + 1
      X(LL) = X(I)
      Y(LL) = Y(I)
      Z(LL) = Z(I)
      IBC(LL) = IBC(I)
      BV(LL) = BV(I)
  320 CONTINUE
C
      NUMNP = LL
C
C           UPDATE THE ELEMENT DATA.
C
      DO 340 I = 1, NUMEL
      DO 330 J = 1, 8
      INOD = IX(J, I)
      IX(J, I) = NOD(INOD)
  330 CONTINUE
  340 CONTINUE
C
  900 CONTINUE
C
      CLOSE (2)
C
 1000 CONTINUE
C
      RETURN
      END
```

After the EAGLE data is read into memory, the element connectivity data is then created. Next, duplicate nodes are removed by considering distance squared between two given node points. When nodes have been removed, the

remaining node points are renumbered so they are consecutively numbered. With this done, the element connectivity must then be modified to reflect the new node numbers.

Subroutine DONE

Subroutine DONE closes all files that are still open, and its listing is given.

```
      SUBROUTINE DONE
C
C
C         THIS SUBROUTINE WRAPS THINGS UP.
C
C
      CLOSE (7)
      CLOSE (8)
      CLOSE (9)
C
      RETURN
      END
```

Subroutine OUTFEM

Subroutine OUTFEM writes the REM grid in 3-D FEM seepage/groundwater format. Its listing is as follows:

```
      SUBROUTINE OUTFEM(IBL)
C
C
C         THIS SUBROUTINE OUTPUTS A FILE FOR THE 3-D SEEPAGE
C         GROUNDWATER FEM PROGRAM.
C
C
      PARAMETER (ND = 1000)
      COMMON /OUTPUT/ X(ND), Y(ND), Z(ND), IBC(ND), BV(ND),
     & IX(9, ND), THETA(ND), PHI(ND), PSI(ND), NUMNP, NUMEL,
     & NUMMAT, NODVEL
      CHARACTER TITLE * 80
      DIMENSION JNODE(4)
C
C         DO TITLE.
```

```
C
      PRINT*, ' '
      PRINT*, 'TITLE CARD?'
      READ 100, TITLE
  100 FORMAT(A80)
      WRITE (2, 100) TITLE
C
C         DO NUMBER OF NODE POINTS, NUMBER OF ELEMENTS,
C         NUMBER OF DIFFERENT MATERIALS, NUMBER OF DISCHARGE
C         VELOCITY CARDS, AND DATUM.
C
      PRINT*, ' '
      PRINT*, 'DATUM?'
      READ (*, *) DATUM
      WRITE (2, 200) NUMNP, NUMEL, NUMMAT, NODVEL, DATUM
  200 FORMAT(4I5, F10.2)
C
C         ADJUST BOUNDARY CONDITION DATA FOR IBC = 2 AND
C         IBC = 12.
C
      DO 205 I = 1, NUMNP
      IF ((IBC(I).EQ.2).OR.(IBC(I).EQ.12)) BV(I) = BV(I) -
     & DATUM
  205 CONTINUE
C
C         DO MATERIAL DATA.
C
      DO 250 I = 1, NUMMAT
      PRINT 210, I
  210 FORMAT(/ ' K1, K2, AND K3 FOR MATERIAL TYPE', I3,
     & ' ?')
      READ (*, *) FK1, FK2, FK3
      WRITE (2, 220) I, FK1, FK2, FK3
  220 FORMAT(I5, 3E10.3)
  250 CONTINUE
C
C         DO NODES.
C
      IF (NUMNP.EQ.0) GO TO 1000
      DO 400 I = 1, NUMNP
      WRITE (2, 300) I, IBC(I), X(I), Y(I), Z(I), BV(I)
  300 FORMAT(2I5, 4F10.2)
  400 CONTINUE
C
C         DO ELEMENTS.
C
      DO 600 J = 1, NUMEL
      WRITE (2, 500) J, (IX(I, J), I = 1, 9), THETA(J)
     & PHI(J), PSI(J)
  500 FORMAT(10I5, 3F10.2)
  600 CONTINUE
C
```

```
C          DO DISCHARGE VELOCITY DATA.
C
      IF (NODVEL.EQ.0) GO TO 900
      REWIND 9
      DO 800 J = 1, NODVEL
      READ (9) JNODE, BCVAL
      WRITE (2, 700) JNODE, BCVAL
  700 FORMAT(4I5, F10.2)
  800 CONTINUE
C
C          DO GRID DIMENSION DATA.
C
  900 WRITE (3, 200) IBL
      DO 910 I = 1, IBL
      READ (7, REC = I) I2, J2, K2, NEL
      WRITE (3, 200) I2, J2, K2
  910 CONTINUE
C
 1000 RETURN
      END
```

Additional information is obtained as needed as the output process is done. This includes a title line, the datum, and material data. Notice that the head values are corrected by the datum if boundary condition 2 or 12 is specified.

Function NONBLK

This function counts the number of characters in a file name so the extensions, .eg1, .eg2, .dim, .sol, etc. can be properly attached. Its listing is now given:

```
      FUNCTION NONBLK(FILENM)
C
C
C          THIS FUNCTION COMPUTES THE NUMBER OF NON-BLANK
C          CHARACTERS IN THE FILE NAME FILENM.  IT WORKS
C          BY LOOKING FOR THE FIRST BLANK CHARACTER.
C
C
      CHARACTER FILENM * (*)
C
```

```
      N = 0
      DO 100 I = 1, 100
      IF (FILENM(I : I).EQ.' ') GO TO 200
      N = N + 1
  100 CONTINUE
C
  200 NONBLK = N
      RETURN
      END
```

APPENDIX C

CONVERSION TO FAST FORMAT

This appendix contains the subroutine to convert the output from the unstructured finite element grid to the multiblocked structured finite volume grid format used by FAST. Although FAST was written primarily to display computational fluid dynamics (CFD) data, it can also be used to present other phenomena as well. The listing of the subroutine is now given.

```
      SUBROUTINE FASTOT(DATUM)
C
C
C         THIS SUBROUTINE CALCULATES THE POSTPROCESSOR FILE
C         FOR FAST.
C
C
      PARAMETER (NDX = 25000)
      PARAMETER (ND = 1250)
      PARAMETER (ND2 = ND * 2)
      COMMON / GRID1 / NUMNP, NUMEL, NUMMAT, X(NDX), Y(NDX),
     & Z(NDX), FX(NDX), NBC(NDX), FLOW(NDX), HLAST(NDX)
      COMMON / GRID2 / XK1(12), XK2(12), XK3(12), NP(9,
     & NDX), THETA(NDX), PHI(NDX), PSI(NDX)
      COMMON / FREE / FRX(NDX), FRY(NDX), FRZ(NDX),
     & COUNT(NDX)
      COMMON  / BANARG / MBAND, NUMBLK, R(NDX), C(ND2, ND)
      DIMENSION VX(NDX), VY(NDX), VZ(NDX), Q(NDX, 8)
      EQUIVALENCE (FRX, VX), (FRY, VY), (FRZ, VZ), (C, Q)
      DIMENSION IPICK(2, 2, 2)
      DATA IPICK /1, 2, 4, 3, 5, 6, 8, 7/
C
C         COMPUTE MAXIMUM AND MINIMUM HEAD.
C
      HMIN = 1.E30
      HMAX = - HMIN
      DO 100 I = 1, NUMNP
      HMIN = AMIN1(R(I) - DATUM, HMIN)
      HMAX = AMAX1(R(I) - DATUM, HMAX)
  100 CONTINUE
      RDH = 1. / (HMAX - HMIN)
C
C         SET AERODYNAMIC DATA.
C
      FSMACH = HMIN
      ALPHA =  HMAX
      RE = 1.
```

```
      TIME = 0.
C
C         WRITE NUMBER OF BLOCKS.
C
      READ (3, *) IBL
      WRITE (12, 130) IBL
      WRITE (13, 130) IBL
  130 FORMAT(3I5)
C
C         WRITE GRID DIMENSION DATA.
C
      DO 150 N = 1, IBL
      READ (3, *) I2, J2, K2
      WRITE (12, 130) I2, J2, K2
      WRITE (13, 130) I2, J2, K2
  150 CONTINUE
C
C         WRITE SOLUTION FOR ALL GRIDS.
C
      REWIND 3
      READ (3, *) IBL
      NEL = 0
C
      DO 1000 N = 1, IBL
C
      READ (3, *) I2, J2, K2
      IROW = I2 - 1
      IPLANE = (J2 - 1) * IROW
C
C         WRITE PRELIMINARY DATA.
C
      WRITE (12, 300) FSMACH, ALPHA, RE, TIME
  300 FORMAT(6(1X, E11.5))
C
C         ACCUMULATE Q VECTOR WITH RESULTS IN IJK ORDER.
C
      INOD = 0
C
      DO 800 K = 1, K2
C
      KK = 1
      IF (K.EQ.K2) KK = 2
C
      DO 700 J = 1, J2
C
      JJ = 1
      IF (J.EQ.J2) JJ = 2
C
      DO 600 I = 1, I2
C
      II = 1
      IF (I.EQ.I2) II = 2
```

```
C
C          GET NODE NUMBER.
C
      MM = (K - KK) * IPLANE + (J - JJ) * IROW + I - II + 1
     & + NEL
      IPOS = IPICK(II, JJ, KK)
      NN = NP(IPOS, MM)
C
C          FOR NODES ABOVE THE FREE SURFACE SET HEAD EQUAL TO
C          ELEVATION.
C
      H = R(NN)
      IF (H.LT.Z(NN)) H = Z(NN)
C
C          NORMALIZE HEAD BETWEEN 1 AND 2.
C
      HN = (H - DATUM - HMIN) * RDH + 1.
C
C          COMPUTE Q.
C
      INOD = INOD + 1
      Q(INOD, 1) = HN
      Q(INOD, 2) = VX(NN) * HN
      Q(INOD, 3) = VY(NN) * HN
      Q(INOD, 4) = VZ(NN) * HN
      Q(INOD, 5) = H - DATUM
C
C          USE THE LAST THREE POSITIONS FOR Q FOR THE NEW
C          (X, Y, Z).
C
      Q(INOD, 6) = X(NN)
      Q(INOD, 7) = Y(NN)
      Q(INOD, 8) = Z(NN)
C
  600 CONTINUE
C
  700 CONTINUE
C
  800 CONTINUE
C
C          OUTPUT Q.
C
      WRITE (12, 300) ((Q(I, J), I = 1, INOD), J = 1, 5)
      WRITE (13, 300) ((Q(I, J), I = 1, INOD), J = 6, 8)
C
      NEL = (I2 - 1) * (J2 - 1) * (K2 - 1) + NEL
C
 1000 CONTINUE
C
      CLOSE (7)
C
```

```
      RETURN
      END
```

Because in an unconfined flow problem the original grid is modified, both a file containing the new grid and a file containing the results are written. C programs were written to take the ASCII output files from this subroutine and convert them to binary files (FORTRAN binary WRITE's will not work). The C programs could, of course, be converted to functions and called from FASTOT as well. Because a totally different flow is being modeled, the preliminary data such as free stream Mach Number, angle of attack, and Reynold's Number are used for other things. Mach Number has minimum head stored in its place, angle of attack has maximum head, and Reynold's Number is set to 1. The unconfined flow part of the seepage/groundwater solver is a solution to a system of nonlinear equations rather than using time dependent equations and allowing the computation to converge to a steady-state condition. Therefore, integration time is not known, and it is set to zero.

The remaining data to be written to the output file are components of the Q' vector

$$Q' = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ U \end{pmatrix}$$

where

ρ = density

u = x component of velocity

v = y component of velocity

w = z component of velocity

U = energy

Again, because of the difference in applications, total head instead of density and discharge velocity instead of velocity are used. In fact, head is normalized between one and two as follows:

$$h_n = \frac{h - h_{min}}{h_{max} - h_{min}} + 1$$

where

h_n = normalized head

h_{min} = minimum head

h_{max} = maximum head

This normalization prevented division by zero by FAST in computing the components of velocity

$$u = \frac{Q_2}{Q_1}$$

$$v = \frac{Q_3}{Q_1}$$

$$w = \frac{Q_4}{Q_1}$$

The momentum type components are then written as $h_n\{v\}$, where $\{v\}$ is the discharge velocity vector. Energy is not used, so the original values of head are output instead.

The array IPICK makes it significantly easier to choose one of the eight nodes of a given element on which to write information (see Chapter V under the heading, "Scientific Visualization," and the subheading, "Conversion from Finite Element to Finite Volume Format," for more details).

Note also that all values of the Q' vector are first computed and then each component is written to the output file. This is because FAST format (originally PLOT3D) requires all values of a given component of the Q vector to be written before another component is processed.

APPENDIX D

GRID GENERATION DATA

This appendix gives the data required to generate the elliptic grid for the two-well-in-an-aquifer problem given in Chapter VI. These include two O blocks generated by EAGLE, two plugs generated by EAGLE, and the data to combine and prepare these data into FEM seepage/groundwater format.

Surface Generation Data

The data for the surface generation portion of EAGLE for the first well O grid is now given.

```
$ 'point', point=1, r=175.,-200.,0. $
$ 'point', point=2, r=175.,300.,0. $
$ 'point', point=3, r=-250.,300.,0. $
$ 'point', point=4, r=-250.,-200.,0. $
$ 'point', point=5, r=175.,-200.,120. $
$ 'point', point=6, r=175.,300.,120. $
$ 'point', point=7, r=-250.,300.,120. $
$ 'point', point=8, r=-250.,-200.,120. $
$ 'setnum', segment=1, points=13 $
$ 'setnum', segment=2, points=6 $
$ 'setnum', segment=3, points=25 $
$ 'setnum', segment=4, points=9 $
$ 'setnum', segment=5, points=17 $
$ 'setval', number=1, value=.002 $
$ 'setval', number=2, value=.04 $
$ 'setval', number=3, value=.02 $
$ 'conicur', points=-2, type='circle', radius=1.,
             angle=59.7,-48.8, coreout=5 $
$ 'conicur', points=-2, type='circle', radius=1.,
             angle=129.8,59.7, coreout=6 $
$ 'conicur', points=-2, type='circle', radius=1.,
             angle=218.7,129.8, coreout=7 $
$ 'conicur', points=-2, type='circle', radius=1.,
             angle=-48.8,-141.3, coreout=8 $
$ 'trans', corein=5, origin=0.,0.,80., coreout=49 $
$ 'trans', corein=5, origin=0.,0.,120., coreout=45 $
$ 'getend', corein=49, point='first', end='first' $
$ 'getend', corein=45, point='first', end='last' $
$ 'line', points=-4, space=-2, coreout=13 $
$ 'getend', corein=49, point='first', end='first' $
$ 'getend', corein=5, point='first', end='last' $
$ 'line', points=-5, space=-3 $
$ 'switch', reorder='reverse1' $
$ 'insert', corein=13, coreout=13 $
```

```
$ 'line', points=-3, r1=2, r2=6, coreout=17 $
$ 'getend', corein=5, point='first', end='first' $
$ 'line', points=-1, r2=2, space=-1, coreout=2 $
$ 'trans', corein=2, origin=0.,0.,120., coreout=26 $
$ 'edgecur', edge='lower1', corein=13 $
$ 'edgecur', edge='upper1', corein=17 $
$ 'edgecur', edge='lower2', corein=2 $
$ 'edgecur', edge='upper2', corein=26 $
$ 'transur', coreout=51 $
$  'current', corein=51, coreout=52 $
$ 'bouncur', corein=13 $
$ 'rotate', angpts=-2, angle=0.,-108.5, axcos=0.,0.,1. $
$ 'switch', reorder='switch', coreout=41 $
$ 'bouncur', corein=13 $
$ 'rotate', angpts=-2, angle=-108.5,-201., axcos=0.,0.,1. $
$ 'switch', reorder='switch', coreout=42 $
$ 'bouncur', corein=13 $
$ 'rotate', angpts=-2, angle=-201.,-289.9, axcos=0.,0.,1. $
$ 'switch', reorder='switch', coreout=43 $
$ 'bouncur', corein=13 $
$ 'rotate', angpts=-2, angle=70.1,0., axcos=0.,0.,1. $
$ 'switch', reorder='switch', coreout=44 $
$ 'current', corein=41 $
$ 'insert', corein=42 $
$ 'insert', corein=43 $
$ 'insert', corein=44, coreout=53 $
$ 'line', points=-3, r1=1, r2=5, coreout=20 $
$ 'line', points=-3, r1=2, r2=6, coreout=17 $
$ 'line', points=-3, r1=3, r2=7, coreout=18 $
$ 'line', points=-3, r1=4, r2=8, coreout=19 $
$ 'line', points=-2, r1=2, r2=1, coreout=9 $
$ 'line', points=-2, r1=3, r2=2, coreout=10 $
$ 'line', points=-2, r1=4, r2=3, coreout=11 $
$ 'line', points=-2, r1=1, r2=4, coreout=12 $
$ 'line', points=-2, r1=6, r2=5, coreout=22 $
$ 'line', points=-2, r1=7, r2=6, coreout=23 $
$ 'line', points=-2, r1=8, r2=7, coreout=24 $
$ 'line', points=-2, r1=5, r2=8, coreout=21 $
$ 'edgecur', edge='lower1', corein=17 $
$ 'edgecur', edge='upper1', corein=20 $
$ 'edgecur', edge='lower2', corein=9 $
$ 'edgecur', edge='upper2', corein=22 $
$ 'transur', coreout=41 $
$ 'edgecur', edge='lower1', corein=20 $
$ 'edgecur', edge='upper1', corein=19 $
$ 'edgecur', edge='lower2', corein=12 $
$ 'edgecur', edge='upper2', corein=21 $
$ 'transur', coreout=42 $
$ 'edgecur', edge='lower1', corein=19 $
$ 'edgecur', edge='upper1', corein=18 $
$ 'edgecur', edge='lower2', corein=11 $
$ 'edgecur', edge='upper2', corein=24 $
```

```
$ 'transur', coreout=43 $
$ 'edgecur', edge='lower1', corein=18 $
$ 'edgecur', edge='upper1', corein=17 $
$ 'edgecur', edge='lower2', corein=10 $
$ 'edgecur', edge='upper2', corein=23 $
$ 'transur', coreout=44 $
$ 'current', corein=41 $
$ 'insert', corein=42 $
$ 'insert', corein=43 $
$ 'insert', corein=44, coreout=54 $
$ 'getend', corein=5, point='last', end='first' $
$ 'line', points=-1, r2=1, space=-1, coreout=1 $
$ 'getend', corein=7, point='last', end='first' $
$ 'line', points=-1, r2=3, space=-1, coreout=3 $
$ 'getend', corein=8, point='last', end='first' $
$ 'line', points=-1, r2=4, space=-1, coreout=4 $
$ 'edgecur', edge='lower1', corein=2 $
$ 'edgecur', edge='upper1', corein=1 $
$ 'edgecur', edge='lower2', corein=5 $
$ 'edgecur', edge='upper2', corein=9 $
$ 'transur', coreout=41 $
$ 'edgecur', edge='lower1', corein=1 $
$ 'edgecur', edge='upper1', corein=4 $
$ 'edgecur', edge='lower2', corein=8 $
$ 'edgecur', edge='upper2', corein=12 $
$ 'transur', coreout=42 $
$ 'edgecur', edge='lower1', corein=4 $
$ 'edgecur', edge='upper1', corein=3 $
$ 'edgecur', edge='lower2', corein=7 $
$ 'edgecur', edge='upper2', corein=11 $
$ 'transur', coreout=43 $
$ 'edgecur', edge='lower1', corein=3 $
$ 'edgecur', edge='upper1', corein=2 $
$ 'edgecur', edge='lower2', corein=6 $
$ 'edgecur', edge='upper2', corein=10 $
$ 'transur', coreout=44 $
$ 'current', corein=41 $
$ 'insert', corein=42 $
$ 'insert', corein=43 $
$ 'insert', corein=44, coreout=55 $
$ 'trans', corein=55, origin=0.,0.,120., coreout=56 $
$ 'trans', corein=51,-56, origin=250.,200.,0.,
           coreout=51,-56 $
$ 'combine', content='yes', corein=51,-56, fileout=1 $
$ 'combine', head='yes', form='e', triad='yes',
             corein=51,-56, fileout=2 $
$ 'combine', form='plot3d', corein=51,-56, fileout=3,
             filnam='aq3.so' $
$ 'end' $
```

Figure 79 shows the line segment numbers.

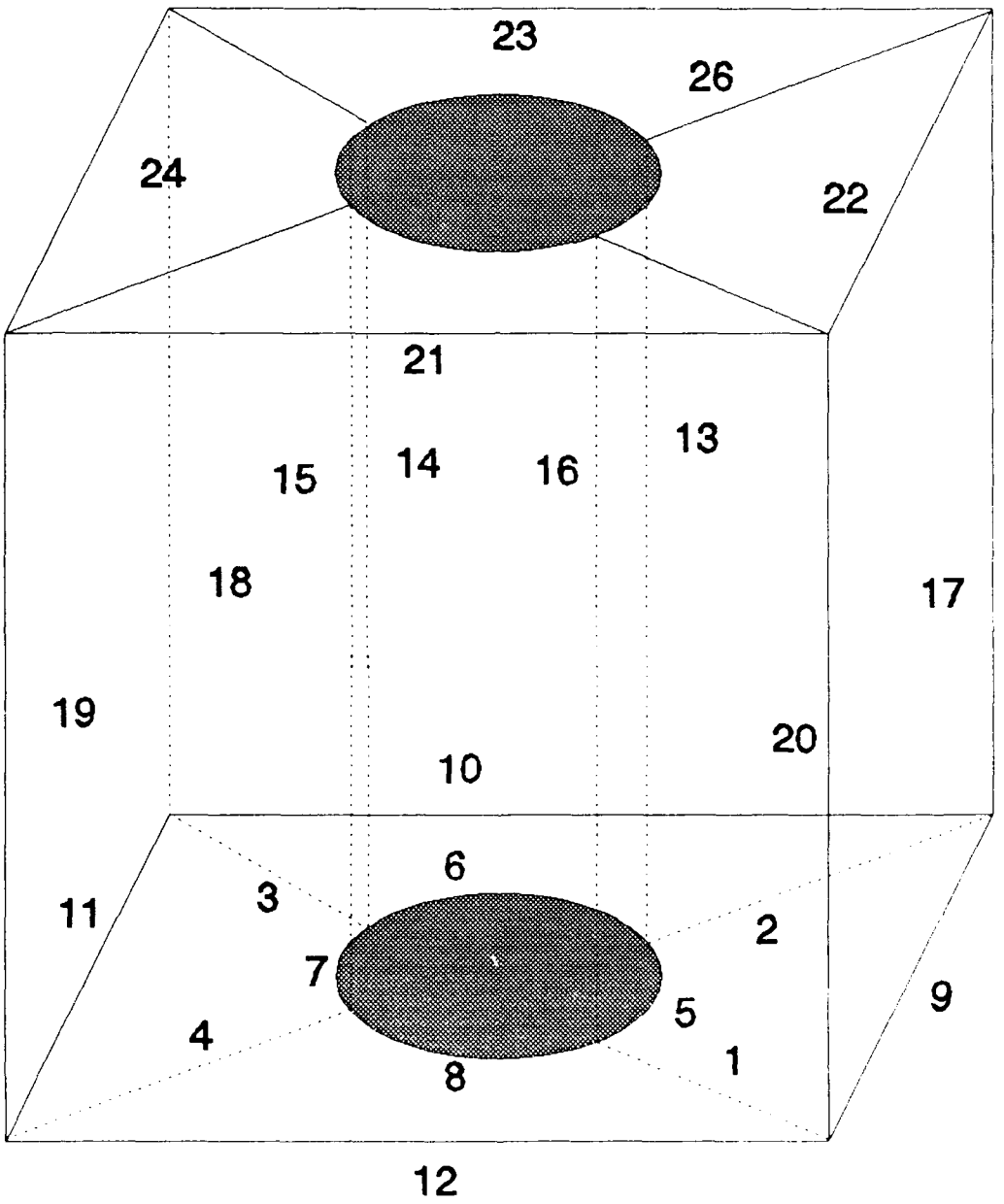


Figure 79. Surface Generation Line Numbers

The very versatile thing about the approach taken is that once the first piece is configured, the second piece of its type can be created by changing the coordinates and a few parameters set at the beginning of the run-stream.

Bottom surface 55 could have been more efficiently generated by first combining the circular segments into one line segment (lower #2), combining the straight line segments into one segment (upper #2), and taking one segment (say line segment #1) as lower #1 and upper #1 in a single transfinite surface operation. The analogous thing is, in fact, what is done for the grid generation code as surface 51 is the same as surface 52, which is the cut.

The surface generation data for the second well and the two plugs are now given.

Well O Grid No. 2

```
$ 'point', point=1, r=400.,-250.,0. $
$ 'point', point=2, r=400.,250.,0. $
$ 'point', point=3, r=-175.,250.,0. $
$ 'point', point=4, r=-175.,-250.,0. $
$ 'point', point=5, r=400.,-250.,120. $
$ 'point', point=6, r=400.,250.,120. $
$ 'point', point=7, r=-175.,250.,120. $
$ 'point', point=8, r=-175.,-250.,120. $
$ 'setnum', segment=1, points=13 $
$ 'setnum', segment=2, points=6 $
$ 'setnum', segment=3, points=25 $
$ 'setnum', segment=4, points=17 $
$ 'setnum', segment=5, points=9 $
$ 'setval', number=1, value=.002 $
$ 'setval', number=2, value=.02 $
$ 'setval', number=3, value=.04 $
$ 'conicur', points=-2, type='circle', radius=2.,
             angle=32.,-32, coreout=5 $
$ 'conicur', points=-2, type='circle', radius=2.,
             angle=125.,32., coreout=6 $
```

```
$ 'conicur', points=-2, type='circle', radius=2.,
             angle=235.,125., coreout=7 $
$ 'conicur', points=-2, type='circle', radius=2.,
             angle=-32.,-125., coreout=8 $
$ 'trans', corein=5, origin=0.,0.,40., coreout=49 $
$ 'trans', corein=5, origin=0.,0.,120., coreout=45 $
$ 'getend', corein=49, point='first', end='first' $
$ 'getend', corein=45, point='first', end='last' $
$ 'line', points=-4, space=-2, coreout=13 $
$ 'getend', corein=49, point='first', end='first' $
$ 'getend', corein=5, point='first', end='last' $
$ 'line', points=-5, space=-3 $
$ 'switch', reorder='reverse1' $
$ 'insert', corein=13, coreout=13 $
$ 'line', points=-3, r1=2, r2=6, coreout=17 $
$ 'getend', corein=5, point='first', end='first' $
$ 'line', points=-1, r2=2, space=-1, coreout=2 $
$ 'trans', corein=2, origin=0.,0.,120., coreout=26 $
$ 'edgecur', edge='lower1', corein=13 $
$ 'edgecur', edge='upper1', corein=17 $
$ 'edgecur', edge='lower2', corein=2 $
$ 'edgecur', edge='upper2', corein=26 $
$ 'transur', coreout=51 $
$  'current', corein=51, coreout=52 $
$ 'bouncur', corein=13 $
$ 'rotate', angpts=-2, angle=0.,-64., axcos=0.,0.,1. $
$ 'switch', reorder='switch', coreout=41 $
$ 'bouncur', corein=13 $
$ 'rotate', angpts=-2, angle=-64.,-157., axcos=0.,0.,1. $
$ 'switch', reorder='switch', coreout=42 $
$ 'bouncur', corein=13 $
$ 'rotate', angpts=-2, angle=-157.,-267., axcos=0.,0.,1. $
$ 'switch', reorder='switch', coreout=43 $
$ 'bouncur', corein=13 $
$ 'rotate', angpts=-2, angle=93.,0., axcos=0.,0.,1. $
$ 'switch', reorder='switch', coreout=44 $
$ 'current', corein=41 $
$ 'insert', corein=42 $
$ 'insert', corein=43 $
$ 'insert', corein=44, coreout=53 $
$ 'line', points=-3, r1=1, r2=5, coreout=20 $
$ 'line', points=-3, r1=2, r2=6, coreout=17 $
$ 'line', points=-3, r1=3, r2=7, coreout=18 $
$ 'line', points=-3, r1=4, r2=8, coreout=19 $
$ 'line', points=-2, r1=2, r2=1, coreout=9 $
$ 'line', points=-2, r1=3, r2=2, coreout=10 $
$ 'line', points=-2, r1=4, r2=3, coreout=11 $
$ 'line', points=-2, r1=1, r2=4, coreout=12 $
$ 'line', points=-2, r1=6, r2=5, coreout=22 $
$ 'line', points=-2, r1=7, r2=6, coreout=23 $
$ 'line', points=-2, r1=8, r2=7, coreout=24 $
$ 'line', points=-2, r1=5, r2=8, coreout=21 $
```

```
$ 'edgecur', edge='lower1', corein=17 $
$ 'edgecur', edge='upper1', corein=20 $
$ 'edgecur', edge='lower2', corein=9 $
$ 'edgecur', edge='upper2', corein=22 $
$ 'transur', coreout=41 $
$ 'edgecur', edge='lower1', corein=20 $
$ 'edgecur', edge='upper1', corein=19 $
$ 'edgecur', edge='lower2', corein=12 $
$ 'edgecur', edge='upper2', corein=21 $
$ 'transur', coreout=42 $
$ 'edgecur', edge='lower1', corein=19 $
$ 'edgecur', edge='upper1', corein=18 $
$ 'edgecur', edge='lower2', corein=11 $
$ 'edgecur', edge='upper2', corein=24 $
$ 'transur', coreout=43 $
$ 'edgecur', edge='lower1', corein=18 $
$ 'edgecur', edge='upper1', corein=17 $
$ 'edgecur', edge='lower2', corein=10 $
$ 'edgecur', edge='upper2', corein=23 $
$ 'transur', coreout=44 $
$ 'current', corein=41 $
$ 'insert', corein=42 $
$ 'insert', corein=43 $
$ 'insert', corein=44, coreout=54 $
$ 'getend', corein=5, point='last', end='first' $
$ 'line', points=-1, r2=1, space=-1, coreout=1 $
$ 'getend', corein=7, point='last', end='first' $
$ 'line', points=-1, r2=3, space=-1, coreout=3 $
$ 'getend', corein=8, point='last', end='first' $
$ 'line', points=-1, r2=4, space=-1, coreout=4 $
$ 'edgecur', edge='lower1', corein=2 $
$ 'edgecur', edge='upper1', corein=1 $
$ 'edgecur', edge='lower2', corein=5 $
$ 'edgecur', edge='upper2', corein=9 $
$ 'transur', coreout=41 $
$ 'edgecur', edge='lower1', corein=1 $
$ 'edgecur', edge='upper1', corein=4 $
$ 'edgecur', edge='lower2', corein=8 $
$ 'edgecur', edge='upper2', corein=12 $
$ 'transur', coreout=42 $
$ 'edgecur', edge='lower1', corein=4 $
$ 'edgecur', edge='upper1', corein=3 $
$ 'edgecur', edge='lower2', corein=7 $
$ 'edgecur', edge='upper2', corein=11 $
$ 'transur', coreout=43 $
$ 'edgecur', edge='lower1', corein=3 $
$ 'edgecur', edge='upper1', corein=2 $
$ 'edgecur', edge='lower2', corein=6 $
$ 'edgecur', edge='upper2', corein=10 $
$ 'transur', coreout=44 $
$ 'current', corein=41 $
$ 'insert', corein=42 $
```

```
$ 'insert', corein=43 $
$ 'insert', corein=44, coreout=55 $
$ 'trans', corein=55, origin=0.,0.,120., coreout=56 $
$ 'trans', corein=51,-56, origin=600.,250.,0.,
           coreout=51,-56 $
$ 'combine', content='yes', corein=51,-56, fileout=1 $
$ 'combine', head='yes', form='e', triad='yes',
             corein=51,-56, fileout=2 $
$ 'combine', form='plot3d', corein=51,-56, fileout=3,
             filnam='aq4.so' $
$ 'end' $
```

Plug No. 1

```
$ 'point', point=1, r=1.,0.,0. $
$ 'point', point=2, r=0.,1.,0. $
$ 'point', point=3, r=-1.,0.,0. $
$ 'point', point=4, r=0.,-1.,0. $
$ 'point', point=5, r=1.,0.,80. $
$ 'point', point=6, r=0.,1.,80. $
$ 'point', point=7, r=-1.,0.,80. $
$ 'point', point=8, r=0.,-1.,80. $
$ 'setnum', segment=1, points=6 $
$ 'setnum', segment=3, points=17 $
$ 'setval', number=1, value=.02 $
$ 'conicur', points=-1, type='circle', radius=1.,
             angle=-141.3,-48.8, coreout=1 $
$ 'conicur', points=-1, type='circle', radius=1.,
             angle=-48.8,59.7, coreout=2 $
$ 'conicur', points=-1, type='circle', radius=1.,
             angle=129.8,59.7, coreout=3 $
$ 'conicur', points=-1, type='circle', radius=1.,
             angle=218.7,129.8, coreout=4 $
$ 'line', points=-3, r1=5, r2=1, space=-1 $
$ 'switch', reorder='reverse1', coreout=5 $
$ 'bouncur', corein=5 $
$ 'rotate', angpts=-1, angle=-141.3,-48.8, axcos=0.,0.,1. $
$ 'switch', reorder='switch', coreout=23 $
$ 'bouncur', corein=5 $
$ 'rotate', angpts=-1, angle=-48.8,59.7, axcos=0.,0.,1. $
$ 'switch', reorder='switch', coreout=22 $
$ 'bouncur', corein=5 $
$ 'rotate', angpts=-1, angle=129.8,59.7, axcos=0.,0.,1. $
$ 'switch', reorder='switch', coreout=24 $
$ 'bouncur', corein=5 $
$ 'rotate', angpts=-1, angle=218.7,129.8, axcos=0.,0.,1. $
$ 'switch', reorder='switch', coreout=21 $
$ 'edgecur', edge='lower1', corein=4 $
$ 'edgecur', edge='upper1', corein=2 $
$ 'edgecur', edge='lower2', corein=1 $
$ 'edgecur', edge='upper2', corein=3 $
```

```
$ 'transur', coreout=25 $
$ 'trans', origin=0.,0.,80., coreout=26 $
$ 'trans', corein=21,-26, origin=250.,200.,0.,
           coreout=21,-26 $
$ 'combine', content='yes', corein=21,-26, fileout=1 $
$ 'combine', head='yes', form='e', triad='yes',
             corein=21,-26, fileout=2 $
$ 'combine', form='plot3d', corein=21,-26, fileout=3,
             filnam='plug1.so' $
$ 'end' $
```

Plug No. 2

```
$ 'point', point=1, r=2.,0.,0. $
$ 'point', point=2, r=0.,2.,0. $
$ 'point', point=3, r=-2.,0.,0. $
$ 'point', point=4, r=0.,-2.,0. $
$ 'point', point=5, r=2.,0.,40. $
$ 'point', point=6, r=0.,2.,40. $
$ 'point', point=7, r=-2.,0.,40. $
$ 'point', point=8, r=0.,-2.,40. $
$ 'setnum', segment=1, points=6 $
$ 'setnum', segment=3, points=9 $
$ 'setval', number=1, value=.04 $
$ 'conicur', points=-1, type='circle', radius=2.,
             angle=-125.,-32., coreout=1 $
$ 'conicur', points=-1, type='circle', radius=2.,
             angle=-32.,32., coreout=2 $
$ 'conicur', points=-1, type='circle', radius=2.,
             angle=125.,32., coreout=3 $
$ 'conicur', points=-1, type='circle', radius=2.,
             angle=235.,125., coreout=4 $
$ 'line', points=-3, r1=5, r2=1, space=-1 $
$ 'switch', reorder='reverse1', coreout=5 $
$ 'bouncur', corein=5 $
$ 'rotate', angpts=-1, angle=-125.,-32., axcos=0.,0.,1. $
$ 'switch', reorder='switch', coreout=23 $
$ 'bouncur', corein=5 $
$ 'rotate', angpts=-1, angle=-32.,32., axcos=0.,0.,1. $
$ 'switch', reorder='switch', coreout=22 $
$ 'bouncur', corein=5 $
$ 'rotate', angpts=-1, angle=125.,32., axcos=0.,0.,1. $
$ 'switch', reorder='switch', coreout=24 $
$ 'bouncur', corein=5 $
$ 'rotate', angpts=-1, angle=235.,125., axcos=0.,0.,1. $
$ 'switch', reorder='switch', coreout=21 $
$ 'edgecur', edge='lower1', corein=4 $
$ 'edgecur', edge='upper1', corein=2 $
$ 'edgecur', edge='lower2', corein=1 $
$ 'edgecur', edge='upper2', corein=3 $
$ 'transur', coreout=25 $
```

```
$ 'trans', origin=0.,0.,40., coreout=26 $
$ 'trans', corein=21,-26, origin=600.,250.,0.,
           coreout=21,-26 $
$ 'combine', content='yes', corein=21,-26, fileout=1 $
$ 'combine', head='yes', form='e', triad='yes',
             corein=21,-26, fileout=2 $
$ 'combine', form='plot3d', corein=21,-26, fileout=3,
             filnam='plug2.so' $
$ 'end' $
```

Grid Generation Data

The data for the portion of EAGLE that generates the grid from the produced surfaces is the same for each piece, except for the numbers on the faces and the output file names. This data for the first O grid is now given.

```
$ 'initial', kstore='core', accpar='optimum', itmax=100,
             tol=.000001, contyp='initial' $
$ 'file', file=11 , all='yes'$
$ 'block'$
$ 'point', point=1, locat=1,1,1 $
$ 'point', point=2, opoint=1, segment=55, direct=1, ndex=1 $
$ 'point', point=3, opoint=2, segment=55, direct=2, ndex=2 $
$ 'point', point=4, opoint=3, segment=55, direct=-1,
           ndex=1 $
$ 'point', point=5, opoint=1, segment=53, direct=3, ndex=2 $
$ 'point', point=6, opoint=5, segment=56, direct=1, ndex=1 $
$ 'point', point=7, opoint=6, segment=56, direct=2, ndex=2 $
$ 'point', point=8, opoint=7, segment=56, direct=-1,
           ndex=1 $
$ 'size', size=7 $
$ 'segment', segment=51, start=1, end=8, class='fix' $
$ 'segment', segment=52, start=2, end=7, class='fix' $
$ 'segment', segment=53, start=1, end=6, class='fix' $
$ 'segment', segment=54, start=4, end=7, class='fix' $
$ 'segment', segment=55, start=1, end=3, class='fix' $
$ 'segment', segment=56, start=5, end=7, class='fix' $
$ 'store', file=14, outer='plot3d', filnam='aq3.eg1' $
$ 'end' $
$ 'error'$
$ 'end' $
```

The point and segment (face) numbers are shown in the topological configuration given in Figure 80.

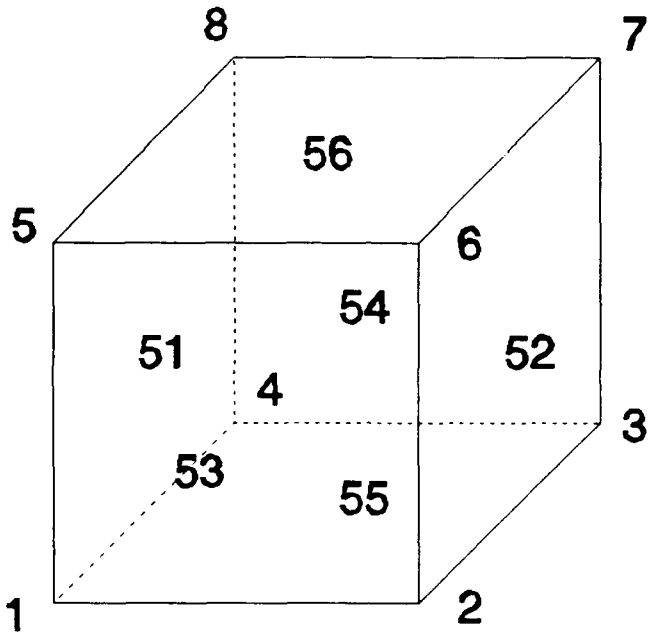


Figure 80. Topology for Grid Program

FEM Grid Preparation Data

The question-and-answer sequence for the program to combine the four separate grids into the proper FEM format is now given.

```
COMMAND?
combine

NUMBER OF FILES
4

FILE NAME NO.  1 ?
aq3.eg1

FOR BLOCK  1 MATERIAL TYPE NUMBER, THETA, PHI, AND PSI?
1 0 0 0
```

```
REMOVING DUPLICATE NODES, N =    1000    NUMNP =            0
REMOVING DUPLICATE NODES, N =    2000    NUMNP =            0
REMOVING DUPLICATE NODES, N =    3000    NUMNP =            0
REMOVING DUPLICATE NODES, N =    4000    NUMNP =            0
REMOVING DUPLICATE NODES, N =    5000    NUMNP =            0
REMOVING DUPLICATE NODES, N =    6000    NUMNP =            0

FILE NAME NO.  2 ?
q4.eg1

FOR BLOCK  1 MATERIAL TYPE NUMBER, THETA, PHI, AND PSI?
1 0 0 0

REMOVING DUPLICATE NODES, N =    1000    NUMNP =         6500
REMOVING DUPLICATE NODES, N =    2000    NUMNP =         6500
REMOVING DUPLICATE NODES, N =    3000    NUMNP =         6500
REMOVING DUPLICATE NODES, N =    4000    NUMNP =         6500
REMOVING DUPLICATE NODES, N =    5000    NUMNP =         6500
REMOVING DUPLICATE NODES, N =    6000    NUMNP =         6500
REMOVING DUPLICATE NODES, N =    7000    NUMNP =         6500
REMOVING DUPLICATE NODES, N =    8000    NUMNP =         6500
REMOVING DUPLICATE NODES, N =    9000    NUMNP =         6500
REMOVING DUPLICATE NODES, N =   10000    NUMNP =         6500
REMOVING DUPLICATE NODES, N =   11000    NUMNP =         6500
REMOVING DUPLICATE NODES, N =   12000    NUMNP =         6500
REMOVING DUPLICATE NODES, N =   13000    NUMNP =         6500

FILE NAME NO.  3 ?
plug3.eg1

FOR BLOCK  1 MATERIAL TYPE NUMBER, THETA, PHI, AND PSI?
1 0 0 0

REMOVING DUPLICATE NODES, N =    1000    NUMNP =        12850
REMOVING DUPLICATE NODES, N =    2000    NUMNP =        12850
REMOVING DUPLICATE NODES, N =    3000    NUMNP =        12850
REMOVING DUPLICATE NODES, N =    4000    NUMNP =        12850
REMOVING DUPLICATE NODES, N =    5000    NUMNP =        12850
REMOVING DUPLICATE NODES, N =    6000    NUMNP =        12850
REMOVING DUPLICATE NODES, N =    7000    NUMNP =        12850
REMOVING DUPLICATE NODES, N =    8000    NUMNP =        12850
REMOVING DUPLICATE NODES, N =    9000    NUMNP =        12850
REMOVING DUPLICATE NODES, N =   10000    NUMNP =        12850
REMOVING DUPLICATE NODES, N =   11000    NUMNP =        12850
REMOVING DUPLICATE NODES, N =   12000    NUMNP =        12850
REMOVING DUPLICATE NODES, N =   13000    NUMNP =        12850

FILE NAME NO.  4 ?
plug4.eg1

FOR BLOCK  1 MATERIAL TYPE NUMBER, THETA, PHI, AND PSI?
1 0 0 0
```

```
REMOVING DUPLICATE NODES, N =    1000    NUMNP =       13122
REMOVING DUPLICATE NODES, N =    2000    NUMNP =       13122
REMOVING DUPLICATE NODES, N =    3000    NUMNP =       13122
REMOVING DUPLICATE NODES, N =    4000    NUMNP =       13122
REMOVING DUPLICATE NODES, N =    5000    NUMNP =       13122
REMOVING DUPLICATE NODES, N =    6000    NUMNP =       13122
REMOVING DUPLICATE NODES, N =    7000    NUMNP =       13122
REMOVING DUPLICATE NODES, N =    8000    NUMNP =       13122
REMOVING DUPLICATE NODES, N =    9000    NUMNP =       13122
REMOVING DUPLICATE NODES, N =   10000    NUMNP =       13122
REMOVING DUPLICATE NODES, N =   11000    NUMNP =       13122
REMOVING DUPLICATE NODES, N =   12000    NUMNP =       13122
REMOVING DUPLICATE NODES, N =   13000    NUMNP =       13122

FINAL NUMBER ON NODES = 13266

COMMAND?
bandwidth

INITIAL MAXIMUM DIFFERENCE = 12881

ITERATION =      1          BEST MAXIMUM DIFFERENCE =  1436
ITERATION =      7          BEST MAXIMUM DIFFERENCE =  1413
ITERATION =      8          BEST MAXIMUM DIFFERENCE =  1375
ITERATION =      9          BEST MAXIMUM DIFFERENCE =  1331
ITERATION =     10          BEST MAXIMUM DIFFERENCE =  1288
ITERATION =     11          BEST MAXIMUM DIFFERENCE =  1252
ITERATION =     12          BEST MAXIMUM DIFFERENCE =  1222
ITERATION =     13          BEST MAXIMUM DIFFERENCE =  1220

COMMAND?
bc

BLOCK NUMBER, I1B, J1B, K1B, I2B, J2B, K2B, IBC, BV?
1 6 13 1 11 13 25 1 0

COMMAND?
bc

BLOCK NUMBER, I1B, J1B, K1B, I2B, J2B, K2B, IBC, BV?
2 6 13 1 11 13 25 1 0

COMMAND?
bc

BLOCK NUMBER, I1B, J1B, K1B, I2B, J2B, K2B, IBC, BV?
1 1 1 17 21 1 25 -2 -.4

COMMAND?
bc
```

```
BLOCK NUMBER, I1B, J1B, K1B, I2B, J2B, K2B, IBC, BV?
2 1 1 9 21 1 25 -2 -.2

COMMAND?
out


OUTPUT FILE NAME FOR SEEPAGE/GROUNDWATER MODEL (WITHOUT
EXTENSIONS)?
aqbs

TITLE CARD?
Two Wells in an Aquifer

DATUM?
120

K1, K2, AND K3 FOR MATERIAL TYPE  1 ?
.1 .1 .1

COMMAND?
end
```

FEM Grid

Portions of the FEM grid data are now given.

```
Two Wells in an Aquifer
 13266 12120    1  480    120.00
    1       0.10       0.10       0.10
    1     0    249.00    199.95      0.00      0.00
    2     0    248.50    199.95      0.00      0.00
    3     0    249.07    199.64      0.00      0.00
    4     0    249.00    199.95      8.07      0.00
    5     0    249.03    200.25      0.00      0.00
    6     0    249.36    199.87      0.00      0.00
    7     0    248.57    199.44      0.00      0.00
    8     0    248.36    199.94      8.11      0.00
    9     0    248.53    200.45      0.00      0.00
   10     0    247.25    199.95      0.00      0.00
                             .
                             .
                             .
13261     0    602.70    250.97    111.93      0.00
13262    -1    601.89    250.66    120.00      0.00
13263     0    602.78    250.32    120.00      0.00
13264     0    604.78    250.57    120.00      0.00
13265    -1    601.70    251.06    120.00      0.00
13266     0    602.68    250.96    120.00      0.00
```

```
    1   394   287   395   469   493   378   505   568    1
          0.00        0.00        0.00
    2   287   200   288   395   378   278   385   505    1
          0.00        0.00        0.00
    3   200   132   201   288   278   197   285   385    1
          0.00        0.00        0.00
    4   132   179   263   201   197   262   366   285    1
          0.00        0.00        0.00
    5   179   251   290   263   262   354   398   366    1
          0.00        0.00        0.00
    6   251   169   202   290   354   252   291   398    1
          0.00        0.00        0.00
    7   169   106   133   202   252   170   203   291    1
          0.00        0.00        0.00
    8   106    60    81   133   170   107   134   203    1
          0.00        0.00        0.00
    9    60    29    44    81   107    61    82   134    1
          0.00        0.00        0.00
   10    29    11    20    44    61    30    45    82    1
          0.00        0.00        0.00
                              .
                              .
                              .
12113  9794 10100 10371 10080 10099 10388 10646 10370    1
          0.00        0.00        0.00
12114 10100 10389 10647 10371 10388 10662 10909 10646    1
          0.00        0.00        0.00
12115 10389 10663 10910 10647 10662 10923 11163 10909    1
          0.00        0.00        0.00
12116  9447  9772 10010  9687  9771 10079 10309 10003    1
          0.00        0.00        0.00
12117  9772 10080 10315 10010 10079 10370 10598 10309    1
          0.00        0.00        0.00
12118 10080 10371 10603 10315 10370 10646 10872 10598    1
          0.00        0.00        0.00
12119 10371 10647 10876 10603 10646 10909 11134 10872    1
          0.00        0.00        0.00
12120 10647 10910 11137 10876 10909 11163 11385 11134    1
        0.          0.          0.
 4168  3860  4176  4500     -0.40
 3860  3576  3878  4176     -0.40
 3576  3303  3594  3878     -0.40
 3303  3535  3836  3594     -0.40
 3535  3828  4142  3836     -0.40
 3828  3529  3829  4142     -0.40
 3529  3240  3530  3829     -0.40
 3240  2958  3241  3530     -0.40
                              .
                              .
                              .
12885 12775 12895 12988     -0.20
12775 12892 12993 12895     -0.20
```

12892	12991	13073	12993	-0.20
12991	13045	13110	13073	-0.20
13045	13117	13166	13110	-0.20
13117	13172	13208	13166	-0.20
13172	13213	13238	13208	-0.20
13213	13242	13257	13238	-0.20
13242	13260	13265	13257	-0.20

WATERWAYS EXPERIMENT STATION REPORTS PUBLISHED UNDER THE COMPUTER-AIDED STRUCTURAL ENGINEERING (CASE) PROJECT

	Title	Date
Technical Report K-78-1	List of Computer Programs for Computer-Aided Structural Engineering	Feb 1978
Instruction Report O-79-2	User's Guide: Computer Program with Interactive Graphics for Analysis of Plane Frame Structures (CFRAME)	Mar 1979
Technical Report K-80-1	Survey of Bridge-Oriented Design Software	Jan 1980
Technical Report K-80-2	Evaluation of Computer Programs for the Design/Analysis of Highway and Railway Bridges	Jan 1980
Instruction Report K-80-1	User's Guide: Computer Program for Design/Review of Curvilinear Conduits/Culverts (CURCON)	Feb 1980
Instruction Report K-80-3	A Three-Dimensional Finite Element Data Edit Program	Mar 1980
Instruction Report K-80-4	A Three-Dimensional Stability Analysis/Design Program (3DSAD) Report 1: General Geometry Module Report 3: General Analysis Module (CGAM) Report 4: Special-Purpose Modules for Dams (CDAMS)	 Jun 1980 Jun 1982 Aug 1983
Instruction Report K-80-6	Basic User's Guide: Computer Program for Design and Analysis of Inverted-T Retaining Walls and Floodwalls (TWDA)	Dec 1980
Instruction Report K-80-7	User's Reference Manual: Computer Program for Design and Analysis of Inverted-T Retaining Walls and Floodwalls (TWDA)	Dec 1980
Technical Report K-80-4	Documentation of Finite Element Analyses Report 1: Longview Outlet Works Conduit Report 2: Anchored Wall Monolith, Bay Springs Lock	 Dec 1980 Dec 1980
Technical Report K-80-5	Basic Pile Group Behavior	Dec 1980
Instruction Report K-81-2	User's Guide: Computer Program for Design and Analysis of Sheet Pile Walls by Classical Methods (CSHTWAL) Report 1: Computational Processes Report 2: Interactive Graphics Options	 Feb 1981 Mar 1981
Instruction Report K-81-3	Validation Report: Computer Program for Design and Analysis of Inverted-T Retaining Walls and Floodwalls (TWDA)	Feb 1981
Instruction Report K-81-4	User's Guide: Computer Program for Design and Analysis of Cast-in-Place Tunnel Linings (NEWTUN)	Mar 1981
Instruction Report K-81-6	User's Guide: Computer Program for Optimum Nonlinear Dynamic Design of Reinforced Concrete Slabs Under Blast Loading (CBARCS)	Mar 1981
Instruction Report K-81-7	User's Guide: Computer Program for Design or Investigation of Orthogonal Culverts (CORTCUL)	Mar 1981
Instruction Report K-81-9	User's Guide: Computer Program for Three-Dimensional Analysis of Building Systems (CTABS80)	Aug 1981
Technical Report K-81-2	Theoretical Basis for CTABS80: A Computer Program for Three-Dimensional Analysis of Building Systems	Sep 1981
Instruction Report K-82-6	User's Guide: Computer Program for Analysis of Beam-Column Structures with Nonlinear Supports (CBEAMC)	Jun 1982

(Continued)

WATERWAYS EXPERIMENT STATION REPORTS PUBLISHED UNDER THE COMPUTER-AIDED STRUCTURAL ENGINEERING (CASE) PROJECT

(Continued)

	Title	Date
Instruction Report K-82-7	User's Guide: Computer Program for Bearing Capacity Analysis of Shallow Foundations (CBEAR)	Jun 1982
Instruction Report K-83-1	User's Guide: Computer Program with Interactive Graphics for Analysis of Plane Frame Structures (CFRAME)	Jan 1983
Instruction Report K-83-2	User's Guide: Computer Program for Generation of Engineering Geometry (SKETCH)	Jun 1983
Instruction Report K-83-5	User's Guide: Computer Program to Calculate Shear, Moment, and Thrust (CSMT) from Stress Results of a Two-Dimensional Finite Element Analysis	Jul 1983
Technical Report K-83-1	Basic Pile Group Behavior	Sep 1983
Technical Report K-83-3	Reference Manual: Computer Graphics Program for Generation of Engineering Geometry (SKETCH)	Sep 1983
Technical Report K-83-4	Case Study of Six Major General-Purpose Finite Element Programs	Oct 1983
Instruction Report K-84-2	User's Guide: Computer Program for Optimum Dynamic Design of Nonlinear Metal Plates Under Blast Loading (CSDOOR)	Jan 1984
Instruction Report K-84-7	User's Guide: Computer Program for Determining Induced Stresses and Consolidation Settlements (CSETT)	Aug 1984
Instruction Report K-84-8	Seepage Analysis of Confined Flow Problems by the Method of Fragments (CFRAG)	Sep 1984
Instruction Report K-84-11	User's Guide for Computer Program CGFAG, Concrete General Flexure Analysis with Graphics	Sep 1984
Technical Report K-84-3	Computer-Aided Drafting and Design for Corps Structural Engineers	Oct 1984
Technical Report ATC-86-5	Decision Logic Table Formulation of ACI 318-77, Building Code Requirements for Reinforced Concrete for Automated Constraint Processing, Volumes I and II	Jun 1986
Technical Report ITL-87-2	A Case Committee Study of Finite Element Analysis of Concrete Flat Slabs	Jan 1987
Instruction Report ITL-87-1	User's Guide: Computer Program for Two-Dimensional Analysis of U-Frame Structures (CUFRAM)	Apr 1987
Instruction Report ITL-87-2	User's Guide: For Concrete Strength Investigation and Design (CASTR) in Accordance with ACI 318-83	May 1987
Technical Report ITL-87-6	Finite-Element Method Package for Solving Steady-State Seepage Problems	May 1987
Instruction Report ITL-87-3	User's Guide: A Three Dimensional Stability Analysis/Design Program (3DSAD) Module	Jun 1987
	Report 1: Revision 1: General Geometry	Jun 1987
	Report 2: General Loads Module	Sep 1989
	Report 6: Free-Body Module	Sep 1989

(Continued)

WATERWAYS EXPERIMENT STATION REPORTS PUBLISHED UNDER THE COMPUTER-AIDED STRUCTURAL ENGINEERING (CASE) PROJECT

(Continued)

	Title	Date
Instruction Report ITL-87-4	User's Guide: 2-D Frame Analysis Link Program (LINK2D)	Jun 1987
Technical Report ITL-87-4	Finite Element Studies of a Horizontally Framed Miter Gate Report 1: Initial and Refined Finite Element Models (Phases A, B, and C), Volumes I and II Report 2: Simplified Frame Model (Phase D) Report 3: Alternate Configuration Miter Gate Finite Element Studies–Open Section Report 4: Alternate Configuration Miter Gate Finite Element Studies–Closed Sections Report 5: Alternate Configuration Miter Gate Finite Element Studies–Additional Closed Sections Report 6: Elastic Buckling of Girders in Horizontally Framed Miter Gates Report 7: Application and Summary	Aug 1987
Instruction Report GL-87-1	User's Guide: UTEXAS2 Slope-Stability Package; Volume I, User's Manual	Aug 1987
Instruction Report ITL-87-5	Sliding Stability of Concrete Structures (CSLIDE)	Oct 1987
Instruction Report ITL-87-6	Criteria Specifications for and Validation of a Computer Program for the Design or Investigation of Horizontally Framed Miter Gates (CMITER)	Dec 1987
Technical Report ITL-87-8	Procedure for Static Analysis of Gravity Dams Using the Finite Element Method – Phase 1a	Jan 1988
Instruction Report ITL-88-1	User's Guide: Computer Program for Analysis of Planar Grid Structures (CGRID)	Feb 1988
Technical Report ITL-88-1	Development of Design Formulas for Ribbed Mat Foundations on Expansive Soils	Apr 1988
Technical Report ITL-88-2	User's Guide: Pile Group Graphics Display (CPGG) Post-processor to CPGA Program	Apr 1988
Instruction Report ITL-88-2	User's Guide for Design and Investigation of Horizontally Framed Miter Gates (CMITER)	Jun 1988
Instruction Report ITL-88-4	User's Guide for Revised Computer Program to Calculate Shear, Moment, and Thrust (CSMT)	Sep 1988
Instruction Report GL-87-1	User's Guide: UTEXAS2 Slope-Stability Package; Volume II, Theory	Feb 1989
Technical Report ITL-89-3	User's Guide: Pile Group Analysis (CPGA) Computer Group	Jul 1989
Technical Report ITL-89-4	CBASIN–Structural Design of Saint Anthony Falls Stilling Basins According to Corps of Engineers Criteria for Hydraulic Structures; Computer Program X0098	Aug 1989

(Continued)

WATERWAYS EXPERIMENT STATION REPORTS PUBLISHED UNDER THE COMPUTER-AIDED STRUCTURAL ENGINEERING (CASE) PROJECT

(Concluded)

	Title	Date
Technical Report ITL-89-5	CCHAN–Structural Design of Rectangular Channels According to Corps of Engineers Criteria for Hydraulic Structures; Computer Program X0097	Aug 1989
Technical Report ITL-89-6	The Response-Spectrum Dynamic Analysis of Gravity Dams Using the Finite Element Method; Phase II	Aug 1989
Contract Report ITL-89-1	State of the Art on Expert Systems Applications in Design, Construction, and Maintenance of Structures	Sep 1989
Instruction Report ITL-90-1	User's Guide: Computer Program for Design and Analysis of Sheet Pile Walls by Classical Methods (CWALSHT)	Feb 1990
Technical Report ITL-90-3	Investigation and Design of U-Frame Structures Using Program CUFRBC Volume A: Program Criteria and Documentation Volume B: User's Guide for Basins Volume C: User's Guide for Channels	May 1990
Instruction Report ITL-90-6	User's Guide: Computer Program for Two-Dimensional Analysis of U-Frame or W-Frame Structures (CWFRAM)	Sep 1990
Instruction Report ITL-90-2	User's Guide: Pile Group–Concrete Pile Analysis Program (CPGC) Preprocessor to CPGA Program	Jun 1990
Technical Report ITL-91-3	Application of Finite Element, Grid Generation, and Scientific Visualization Techniques to 2-D and 3-D Seepage and Groundwater Modeling	Sep 1990